

All Questions



<u>Tell Me About</u>	<u>Soft Skill Testing</u>	<u>SDLC-STLC-AGILE</u>	<u>Behavioral</u>
<u>JAVA</u>	<u>SELENIUM</u>	<u>TestNG - JUnit</u>	<u>CUCUMBER</u>
<u>API</u>	<u>SQL</u>	<u>JENKINS – GIT</u> <u>MAVEN</u>	<u>AWS</u> <u>JMETER - Linux</u>
<u>Java Coding</u>	<u>Log4J</u>	Java Codes new	

TELL ME ABOUT



<u>Tell me about YOURSELF</u>	<u>Daily Activities</u>
<u>Tell me about your PROJECT</u>	<u>Role</u>
<u>Tell me about your FRAMEWORK</u>	<u>Team Structure</u>
<u>How you are implementing</u> <u>AGILE in your current project?</u> (Sprint Ceremony)	<u>Main Responsibilities as an SDET</u>

Tell me about yourself



First of all, I would like to thank you for giving me this opportunity and I really appreciate your time.

I've been working in the IT industry for 8 years. Throughout my career I've specialized in automation, but I was also involved and I am very comfortable with manual and back-end testing.

Currently, I am holding an SDET position in my team and my main responsibilities are to design, develop and maintain Test Automation Framework that verifies user stories and system requirements.

I have hands on experience on UI, API and DataBase testing. My role in my company includes running FUNCTIONAL TEST, SMOKE TEST and REGRESSION TEST for UI part and also API Testing for the backend.

- I have specialized in JAVA. I used SELENIUM as a testing tool with CUCUMBER BDD framework. I developed my “testing framework” from scratch based on Page Object Model (POM) and I have worked with MAVEN as a build management tool and SELENIUM with JUnit testing framework. I have also worked with TestNG and Data Driven Testing using Apache POI.

- I use GIT for version control and I also use JIRA for bug tracking and project management tool.

- I have extensive knowledge of SQL queries and I used POSTGRESQL

to interact with a relational database and and JDBC for database testing. I am also comfortable with the Data Driven (DDD) Frameworks.

- I have worked on API testing on my project and I used POSTMAN and REST ASSURED LIBRARY for API testing.

- I achieve continuous integration and schedule my test executions by using Jenkins.

- I'm familiar with Agile ENVIRONMENTS and currently I'm working in a Scrum Team and proficient in all sprint-related Scrum ceremonies.

- I am a certified SCRUM MASTER and OCA certification (ORACLE JAVA PROGRAMMER).

- I am a cross- functional team member. As far as soft skill concerned, I consider myself a positive person, adaptable to changing circumstances,

- I can work well individually and in a team. I am detail oriented, a quick learner, passionate about learning new skills and new technology, and always make sure I meet the deadlines. That's pretty much about myself.

Tell me about your Project



- **My current project is a web-based loan application.**
 - As you can see from my resume, I am currently working in a loan company that works with small companies. I am working for loan authorization department.
 - The application we are responsible for is mainly developed to store the all required information about the customer and share it with the company headquarter. It also generates the details of the status of all pending, denied and authorized loan applications.
 - As an SDET my main responsibly is automating the UI part of the application, but I am also aware of the back end and database part of the application.
 - My role in this project includes running regression, smoke and functional testing. As a cross functional team member, I also I help manual testing when required.

Daily Activities



- My daily activities at work, I go to work early in the morning and check result report of Smoke Test to make sure that environment is up and running and the application is stable or not for the day.
- If something goes wrong, I will send out an email to my team so they can take care of it asap before everyone comes to work, to reach maximum productivity.
- And then I check my email if there are any important tasks or notices, also check my schedule if there are any meetings for the day and also check Jira to review what needs to be done that day in which priority.
- Then I go to attend daily standup meeting at 10:00 am. with my scrum team to talk about what we did yesterday, what we will do today and are there any impediments in my way. This meeting is simply to synchronize our team and it takes about 15 minutes.
- After that, I go back to my desk and start automating test cases from regression suits. And also, I automate test cases from sprint backlog after doing manually if it is passed.
- That is pretty much about my daily activities at work.

Role



- As an automation engineer, I develop my 'testing framework' based on POM (Page Object Model). I performed various types of testing, like; functional testing, smoke testing, regression testing and back-end testing. I am responsible executing Regression test when developers add new functionality to the application or end of the sprints.
- I am also responsible to check report of Smoke Test to make sure that environment is up and running first thing in the morning.
- If there are any issues, I analyze them.
- If it is service issue, I will immediately contact developers.
- If it is about my scripts, I debug my scripts and fix it.
- If it is a bug, I re-produce it and log the defect.
- And also I am using Jira as bug management tool. Once the bug reports fixed by the developers, I re-test it and if it is passed I close it. If the defect is not fixed, I re-open it. Also, as a part of the Agile Scrum Team, I participate in the several walkthroughs meeting for the requirement reviews and provide valuable feedback to the BA. Lastly, I am cross-functional team member that is always willing to help my team in any way to achieve our sprint goal.
- That is pretty much about my role as an automation engineer.

Framework



- My framework is written using Java OOP language. Java is a powerful and robust programming language and selenium with java is an opensource and has a large community that support each other.
- Selenium WebDriver is a library/tool/API which is used to automate the browser, it interacts with the browser. It used for UI automation testing.
- My framework is created as a maven project; maven is used to manage dependencies and also run our tests, as mvn goals from terminal.
- We use Junit as a testing tool. Junit is used to kick off cucumber tests and also do assertions.
- In my framework we are using POM according to which has separate class for every pages of the application.
- My framework uses a singleton pattern to share the WebDriver instance between different classes.
- I have in my framework configuration.properties file to store the important test data, like usernames, passwords, url's, browsers etc..
- In my Utilities package I have reusable utilities which can be used across different classes of my framework, like waits that I need to use always, ConfigurationReader that reads my configuration.properties file, and My Driver that I can choose which browser I will use in this test case...
- (Types of tests) My framework can test the UI, Back-end and API
- We used Cucumber to write test cases, requirements, specifications in a GHERKIN language understandable by non-tech people
- My framework generates detailed HTML and JSON reports with is easy to read and understand to non-technical team members. My reports have details test steps and screenshots for any failures that may occur. It can also do metrics on what percentage is passing, failing, skipped etc.
- We use Git, Github for Version Control.
- As a Continuous Integration/Deployment tool we use Jenkins.

Team Structure



- My team consist of adaptive, cross-functional and self-organized individuals that highly motivated and knowledgeable. We have 11 people in my team.
 - 5 developers (Jacobs, Suzan, Tim, Alejandra, Megane),
 - 3 Testers (Irina, Alex, me)
 - 1 SM (John)
 - 1 PO (Brian)
 - 1 BA (Shaun)



Main Responsibility as an SDET?

- Turn manually executed test scenarios into automatically executed test scenarios via a Selenium, Java and Cucumber testing framework and Gherkin.
- Design and develop test plans that verify user stories and system requirements.
- Develop and automate test cases to ensure what we build meets the highest levels of quality.
- Functional Testing, Regression Testing, Smoke Testing

Agile experience in your most recent project?

- Our sprint is 2 weeks and we have release every 3 sprints as a release cycle. We have 11 people in my team. 5 developers , 3 testers, also 1 SM, 1 BA and 1 PO.
- We start a sprint with Sprint Planning Meeting and we learn the part of the application which we are going to develop. We get general idea than we do Sprint Grooming for giving some estimates for the tasks.
- After sprint starts, we do Daily Standup Meeting everyday morning and we discuss what did we do yesterday, what will we do today and is there any blocker. Just we synchronize info about the sprint.
- End of the sprint, usually we do Sprint Demo/Review Meeting . It is just to show customer what we build throughout the sprint. PO puts feedback. As an SDET in my team, I have done presentation sometimes and go over through the functionalities in the meeting room. Client or stakeholders or business people they ask questions what they don't know or what they want to know.
- After Sprint Demo, we do Sprint Retrospective Meeting. In Sprint Retro, we talk about what was good in last sprint, what kind of mistakes we made. We go over them and make sure that we don't make the same mistakes again. If we did something good , we would continue doing it.
- This is pretty much our Sprint process



SOFT SKILL TESTING

- Software Testing?
- Why we test?
- Test Plan Vs Test Strategy
- Test Plan?
- Test Case?
- When the testing starts?
- Defect Life Cycle?
- Epic, User stories & Tasks?
- What is testing hierarchy?
- Requirement Traceability Matrix (RTM)
- Defect
- What to do when you find a defect?
- If Developer says not a defect, what to do?
- When will you Automate?
- When will you not Automate?
- Debugging vs testing?
- Peer review?
- Who writes test plans and test cases?
- Bug severity vs bug priority
- System testing and integration testing?
- Black box vs white box testing?
- Front End Testing and Back End testing?
- Functional testing & non-functional testing

Software Testing?



- Process of executing a program or application with the intent of find software bugs using functional and automation tools
- Process of validating/verifying a software program/application

Why we test?



- To build bug free application.
- To satisfied end user and client.
- To build great product to generate more revenue.

Is 100% testing possible?



- We can't test the application 100% since there are unlimited scenarios that we can't even imagine.
- Software testing is risk based activity based on priority of the functionality we can test as much as much as possible.
- Even though 100% testing is not possible but I believe 100% customer satisfaction is certainly possible.

Test Plan VS Test Strategy?



- A **Test Plan** is a formal document derived from requirement documents, describing in detail the scope of testing and the different activities performed in testing. Whereas, a **test strategy** is a high-level document describing the way testing will be carried out in an organization.

#	Test Plan	Test Strategy
1	A test plan is derived from Software Requirement Specification (SRS), describing in detail the scope of testing and the different activities performed in testing.	A test strategy is a high-level document describing the way testing is carried out.
2	A test plan is specific to a particular project.	A test strategy is normally for a complete organization. Although it can be specified for a particular project as well.
3	It describes the whole testing activities in detail - the techniques used, schedule, resources etc.	It describes the high-level test design techniques to be used, environment specifications etc.
4	It is prepared by test lead or test manager.	It is generally prepared by the project manager.

Test Plan?



- Test plan is a word document that described the testing scope
 - High level test cycle
 - Defect life cycle
 - Entrance Criteria (defines what all need to start the testing)
 - Exit Criteria (defines what the testing is finished)
- If you don't know where to start and where to finish then your goals are not clear.

Test Case?



- Test case describes the functionality and test steps.
 - Test Case ID
 - Step number
 - Description of the functionality
 - Expected result
 - Actual Result

When the testing starts?



- Testing starts from testing the requirements (not after the coding phase which seems like the most likely answer.)
- We have to make sure the requirement is correct in first place.
- With the wrong requirement it is impossible to build bug free application.

Defect Life Cycle?



- New
- Assigned
- Open
- Fixed
- Retested
- Close

Epic, User stories & Tasks?



- **User Stories:** User Stories defines the actual business requirement. Generally created by Business owner
- **Task:** To accomplish the business requirements development team create tasks
- **Epic:** A group of related user stories is called an Epic

What is testing hierarchy?



- **Unit testing** – Developers test each module or block of code during development.
- **Component Testing** – Component is a standalone functionality that can work by itself. Ex. Amazon Buyer Functionality, Seller Functionality, Prime Video Functionality.
- **Integration Testing** – Combine all of the Functionalities. When I integrate them, can I still use all of the functions? Make sure they all still work.
- **System Testing** – EndtoEnd testing. Test everything from beginning to end.
- **Acceptance Testing** – Hire a UAT (User Acceptance Testing) Team or Business Analyst can also do Acceptance Testing.
After testing has been complete you have to get another team to do acceptance testing so they can confirm the QA teams testing was successful and have the product ready for the customer.

Requirement Traceability Matrix (RTM)



- RTM is used to make sure that all test cases cover the requirement or not. It is like excel sheet. We can also say that RTM is an document which shows all the acceptance criteria and the user stories are covered by the Test Cases (Scenario)

Defect



- When expected result does not match actual result it is a defect.

What to do when you find a defect?



- If I find a defect, before report it I re-produce the bug that I need to make sure that is a valid defect.
- If it is a small issue, I will go to the developer desk, and he can fix it right away.
- If it is a big issue, then I open my JIRA and log the defect.
- If I am not sure it is bug or not I will talk to SME (subject matter expert it means the person who knows the application better than anyone).

If Developer says not a defect, what to do?



- I always make sure that it is a real defect that's why I reproduce it.
- I take a screenshots and give all the steps to reproduce the defect
- Actually one of my biggest challenges that I faced in my current project is that...

When will you Automate?



- If it is taking a lot of manual effort. I run at least once manually and after that I automate it. Automation is good for most repetitive functionality.
 - If the test cases are high priority test cases.
 - If the functionality is critical functionality.
 - If the test cases are too long and too difficult to execute. The regression test cases based on the priority.
 - We should automate as much as possible.

When will you not Automate?



- -If functionality keeps changing
 - If functionality is used only once during the entire project
 - Ad-Hoc test can not be automated.
- Captcha can not be automated

Debugging vs testing?



- The main difference between debugging and testing is that debugging is typically conducted by a developer who also fixes errors during the debugging phase. Testing on the other hand, finds errors rather than fixes them. When a tester finds a bug, they usually report it so that a developer can fix it.

Peer review?



- Peer reviews are reviews conducted among people that work on the same team. For example, a test case that was written by one QA engineer may be reviewed by a developer and/or another QA engineer.

Who writes test plans and test cases?



- Test plans are typically written by the quality assurance lead while testers usually write test cases.

Bug severity vs bug priority



- **Bug Severity** refers to the level of impact that the bug has on the application or system while **Bug Priority** refers to the level of urgency in the need for a fix.
- Usually the **severity** is defined in terms of financial loss, damage to environment, company's reputation and loss of life.
- **Priority** of a defect is related to how quickly a bug should be fixed and deployed to live servers.

5 Levels of Severity

1. Critical
2. Major
3. Moderate
4. Minor
5. Cosmetic

System testing and integration testing?



- For system testing, the entire system as a whole is checked,
- whereas for integration testing, the interaction between the individual modules are tested.

Black box vs white box testing?



- **White Box Testing** has many names such as Glass Box, Clear Box, or Structural Testing. It requires the testers to gain code-level perspective, design cases to exploit code and find potential bugs.
- **Black Box Testing** is a standard software testing approach which requires testers to assess the functionality of the software as per the business requirements. They treat the software as a black box and validate as per the end user point of view. It applies to all levels of software testing such as Unit, Integration, System or Acceptance Testing.

Front End Testing and Back End testing?



- Front End Testing is performed on the Graphical User Interface (GUI), whereas Back End Testing involves databases testing. Front end consist of web site look where user can interact whereas in case of back end it is the database which is required to store the data.
- When user enters data in GUI of the front end application, then this entered data is stored in the database. To save this data into the database we write SQL queries.

Functional testing & non-functional testing



- **Functional testing** is a type of testing which verifies that each function of the software application operates in conformance with the requirement specification.
 - -Unit Testing
 - -System Testing
 - -Smoke Testing
 - -User Acceptance Testing
 - -Integration Testing
 - -Regression Testing
- **Non-functional testing** is a type of testing to check non-functional aspects (performance, usability, reliability, etc.) of a software application.
 - -Performance Testing
 - -Stress Testing
 - -Load Testing
 - -Security Testing



SDLC- STLC - AGILE

- STLC vs SDLC
- Validation and verification ?
- Requirement document?
- Where is the requirement coming from?
- The requirement is good or bad?
- Agile
- Agile Framework?
- Waterfall?
- Agile methodology did you use in your previous projects ?
- Relation between Agile and Scrum? What is Scrum?
- Why do we need Agile? Waterfall and Agile?
- Different roles in Scrum?
- Describe a scrum team?
- Scrum of scrums?
- Burn-down and burn-up charts
- Sprint?
- Product backlog & Sprint Backlog?
- Velocity of a sprint and how it is measured?

STLC vs SDLC



- STLC is part of SDLC. It can be said that STLC is a subset of the SDLC set. The complete Verification and Validation of software is done in SDLC, while STLC only does Validation of the system.

SDLC (Software Development Life Cycle)

1. Requirement gathering and analysis
2. Design
3. Coding
4. Testing
5. Deployment
6. Maintenance

STLC (Software Testing Life Cycle)

1. Requirement / Design Analysis
2. Test Planning
 - Test Plan
 - Test Estimation
 - Test Schedule
3. Test Case Development (Designing)
 - Test Cases / Test Scripts / Test Data
 - Requirements Traceability Matrix
4. Test Environment Setup
5. Test Execution
 - Test Results (Incremental)
 - Defect Reports
6. Test Closure Activity (Reporting)
 - Test Results (Final)
 - Test Metrics
 - Test Closure Report

Validation and verification ?



- **Validation** is the process, whether we are building the right product
- **Verification** is the process, to ensure that whether we are building the product right

Requirement document?



- Requirements convey the expectation of users for the software or product. In other words, all the expected functionalities out of the application are documented in terms of “Requirements” and this document is called a Requirement document. It is also called an SRS document, which stands for System Requirement Specification Document.
The SRS document is prepared by the Business Analyst by taking all the requirements for the customer.

Where is the requirement coming from?



- Customers give requirements for the application Talk to the End-users the person that will be using this application the most
- Talk to Partners
- Talk to Domain Experts – coders and developers that have already build this application similar before or someone that is an expert the type of product being built
- Industry Analysts and Information about competitors

The requirement is good or bad?



- Requirement must be (SMART)
Specific User should be able to login. Authorized
- user with valid username and password should be able to login
Measurable User should able to login very fast (in 2 second after clicking login button).
- Attainable
Realistic
Testable

Agile



- Agile is flexible methodology using for Software Development.
- You can go back.
- It keeps on changing.
- Changing is always welcome.

Agile Framework?



- **Role** : PO, SM, Team
- **Ceremonies** : Sprint Planning, Daily Scrum, Sprint Review, Sprint Retro, Grooming Session
- **Artifacts** : Product backlog, Sprint backlog, Burnout chart

Waterfall?



- Waterfall methodology is the sequential method using for Software Development.
- You can not go back and have to finish the phase before you move on.

Agile methodology did you use in your previous projects ?



- I have heard Extreme programming(XP) , Kanban and Scrum.
- But I have only worked with scrum only.

Relation between Agile and Scrum? What is Scrum?



- Agile is the software development methodology that focuses on customer satisfaction by delivery shippable software frequently.
- Scrum is one of the many approaches to implement Agile.
- Scrum is an Agile framework.
- Scrum is suitable for certain type of projects where there are rapidly changing requirements.
- In simple words, Agile is the practice and scrum is the process to following this practice.

Why do we need Agile? Waterfall and Agile?

- Because waterfall methodologies have following disadvantage;
Requirement can not be change or hard to change once document is signed.
- In waterfall before completing the one phase you can't move to the next phase.
For example, before coding phase is completed testing can not be started.
- Customer can't see what they are going to get until very late stage in development life cycle.
It takes longer time to go to the production. By the time product goes to the market it might be outdated already.
- Agile has following advantages:
The change is welcomed. For example after the sprint demo if client does not like something we can take their feedback and improve the product. Requirement change is OK.
Since it is iterative development process, the development team can developed piece of functionality, get feedback and improve next iteration. So the product will be continuously improve.

Different roles in Scrum?



- **Product owner** is actually the stakeholder of the project. He represents the project requirements before the team. He is responsible to have a vision of what to build and convey his detailed vision to the team. He is the starting point of an agile scrum software development project.
- **Scrum team** is formed by the collective contribution of individuals who perform for the accomplishment of a particular project. The team is bound to work for the timely delivery of the requested product.
- **Scrum master** – Scrum master is the leader and the coach for the scrum team who checks whether the scrum team is executing committed tasks properly. He is also responsible to increase the efficiency and productivity of the team so that they can achieve the sprint goal effectively.

Describe a scrum team?



- For me the team is a group of people who are sharing the same goal , moving to the same direction , who trust each other and who will effectively communicate and collaborate with each other to build great product. There should be no star individual but a star team.

Scrum of scrums?



- In case, there are multiple teams involved in the project, scrum of scrums is used to scale the daily stand-up meeting.
- It supports agile teams to collaborate and coordinate their work with other teams.

Burn-down and burn-up charts



- To track the progress of an ongoing project, these charts are used.
- **Burn-up charts** indicate the work that has been completed while
- **Burn-down chart** shows the amount of remaining work in a project.

Sprint?



- In Scrum, the project is divided into Sprints.
- Each Sprint has a specified timeline (2 weeks to 1 month). This timeline will be agreed by a Scrum Team during the Sprint Planning Meeting. Here, User Stories are split into different modules. The end result of every Sprint should be a potentially shippable product.

Product backlog & Sprint Backlog?



- **Product backlog** is maintained by the project owner which contains every feature and requirement of the product.
- **Sprint backlog** can be treated as subset of product backlog which contains features and requirements related to that particular sprint only.

Velocity of a sprint and how it is measured?

- **Velocity** is one of the planning tool used to estimate the speed of the work and time of completion of the project.
- The calculation of velocity is done by reviewing the work team has successfully completed during earlier sprints;
- for example, if the team completed 5 stories during a two-week sprint and each story was worth 3 story points, then the team's velocity is 15 story points per sprint.



BEHAVIORAL QUESTIONS

- Biggest Accomplishment?
- Why did you apply for this position?
- Where do you see yourself 5 years from now?
- Why should we hire you?
- Weakness?
- Strength?
- Challenge you faced during your last project?
- How do you handle stress? Or Conflict?
- Can you start tomorrow?
- What do you do if I hire you?
- How long are you planning to stay?
- Can you work under pressure?
- What do you like the most about testing?
- What to do in case of you have too much work and you can...
- Do you have any question for us?

Biggest Accomplishment?



- I would say is establishing a great trustworthy relationship within the team. If you are asking for technical: When I joined my last project, the application had very less “id” so I had to spend hours to locate one WebPage elements in my POM project so I communicated with developers and other team members and all together we come up with the solution which I got the access to put id in the application by myself. That was great for me it saved my and others time. So instead of spending time to locating elements I spend my time to more creating automation test scripts and executing them.

Why did you apply for this position?



- After looking at the job description, I think it matches my day-to-day activity and my experience.
- I was confident with the job description that's why I applied.
- Also, I have done some research on the company and I am really excited about the company's product and services like...

Where do you see yourself 5 years from now?

- I want to learn as much as possible to be more technical. I would like to see myself SDET. I want to be technically very competitive person 5 years from now.

Why should we hire you?



- I think you should hire the candidate that has the best qualifications for this position. Since I don't know the other candidates, I can represent only myself. I think my experience and technical expertise will bring a lot of values and benefits to the company and the project. I think that's why you should hire me.

Weakness?



- Well, I think my weakness is that whenever I am given some responsibilities and there is a deadline for it, I work day and night, sometimes 7 days a week. This is bad for my family life, the reality is I can not sleep unless I am done with my assignments.

Strength?



- Well, I am very detail oriented person. I have the sense of urgency. I can prioritize my job according to the deadline. I am very much dedicated towards my job. I am honest. I have the skills and expertise in QA process. These are some of my strengths.

Challenge you faced during your last project?



- I think, one of the biggest challenges that I faced with in my current project is that...
- ... everytime I found a bug, the developer disagreed to accept it and most of the time we had to ask BA for clarification. Then I realize the requirement itself was not specific enough, so I understood it differently than the developer. In the Sprint Retro, I said we should spend more time on requirement clarification because you know that is the key to the project success. We did so and this issue was solved. Work on result more. I think, the most important problem is misunderstanding and the lack of communication in the business life. If we come together as a group and discuss it, there is nothing we cannot solve. I'm really grateful and blessed to have been in the team that I was in, because we were able to collaborate and come together to solve the problem.
- The challenge I have faced is locating dynamic elements by retrieving the right HTML code. One of my recent challenge is that another coworker who is also QA had to leave.

How do you handle stress? Or Conflict?



- Nothing is personal. Everyone thinks company's benefits so I would like to explain my concern and his/her explanation makes sense for me. Of course, I can do the things which is most helpful to my company. So, I try to communicate with his/her and I would try to understand the concern. Because everyone have the same goal and wants to get job done successfully. Also in scrum environment we working as a team . I always maintain good communication and relationship with my colleagues. So they trust me and they can communicate with me very easily. . . I always avoid miscommunication and my team believe me every time.

Can you start tomorrow?



- My team won't be happy with me if I leave tomorrow, and I don't think it is professional and I have never done that before. I have to transfer the automation framework knowledge to other team members before I leave.

What do you do if I hire you?



- In first week, you know, I will get done all the paper works, getting the machines and necessary access to the project, databases etc. Then I will have to learn the company culture. I have to learn also more about my projects and my teammates. I think, understanding what the project is doing, it is very important if I want to be more productive.

how long are you planning to stay?



- As long as there is a project to work, I am willing to stay as long as possible.

Can you work under pressure?



- I don't remember any project that I worked had no pressure. Pressure is good thing sometimes. It forces you to work harder and smarter.

What do you like the most about testing?



- Testing is fun job for me because you are very important person to the client and end users. I love testing because as end user I want to buy better product that is piece of art and defect free. Also, I am helping others to make sure their product has top quality.

What to do in case of you have too much work and you can not finish for the deadline?



- When developers don't deploy their code on time, our tester team don't have enough time for completion. And the upper management keeps asking for us for completion. - Some of my team members simply focuses on task completion and not on the test coverage and quality of work. - So, at the Sprint Grooming Meeting, I suggested that we should work very closely with the developer and make sure that we are communicating on daily base. And also, the developers prioritize the important tasks and work on them first. Any scenarios left, would be pushed to the next sprint since it is not as important as the other ones. - Lastly, I try to prioritize my work and follow my test lead and manager whatever they see is more important I start with that.

Do you have any question for us?



- it's important to me that I continually improve and try to achieve excellence in my position and the best way to do this is to continually learn. I'm always trying to learn new things or learn old things better.
- Do you provide trainings, seminars or anything to support the education of your employees?
- What are the next steps in the interview process?



JAVA GENERAL

JDK, JRE and JVM

public static void main (String args[])

String - StringBuilder- StringBuffer-Mutable-Immutable

String Pool and Heap in Java

Classes & Objects

this & Super keywords [this. - this() - Super. – Super()]

METHODS

STATIC Keywords (variable, methods, blocks)

Local Variable - Instance Variable - Static Variable

OVERLOADING Method

OVERRIDING Methods

Overloading vs Overriding

JAVA GENERAL

Constructor

Constructor – Rules for Constructor

Constructor vs Method

Final vs Finalize vs Finally?

Access Modifiers in Java

equals() vs == in Java

Pass by value or Pass by reference

Break vs Continue

Java 7 vs Java 8

Lambda Expression in Java

Thread-safe or Synchronized

Multithreading in Java



JVM: Java Virtual machine(JVM) is an abstract machine. It doesn't physically exist. Whatever Java program we want to run, goes into JVM. And JVM is responsible for loading, verifying and executing the java program. JVM is responsible for converting the byte code to the machine-specific code.

JRE: JRE stands for “Java Runtime Environment”. It physically exists. The Java Runtime Environment provides the minimum requirements such as libraries and Class Loader for executing a Java application on JVM. It consists of the Java Virtual Machine (JVM), core classes, and supporting files. *JRE = JVM + Library Classes*

JDK: The Java Development Kit (JDK) is a software development environment used for developing Java applications. *JDK = JRE + Development tools*. It includes:

- Java Virtual Machine,
- Java Runtime Environment,
- Loader,
- Java compiler,
- Documentation generator
- Archiver (jar),
- Other tools needed in Java development.

public static void main (String[] args) in Java



- public:

public is an access modifier which is used to specify who can access this method. Public means that this method will be accessible by any Class.

- static:

It is a keyword in java which identifies it is class based i.e it can be accessed without creating the instance of a class

- void:

it is the return type of the method void defines the method which will not return any value

- main:

it is the name of the method which is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs

- String args:

it is the parameter passed to the main method

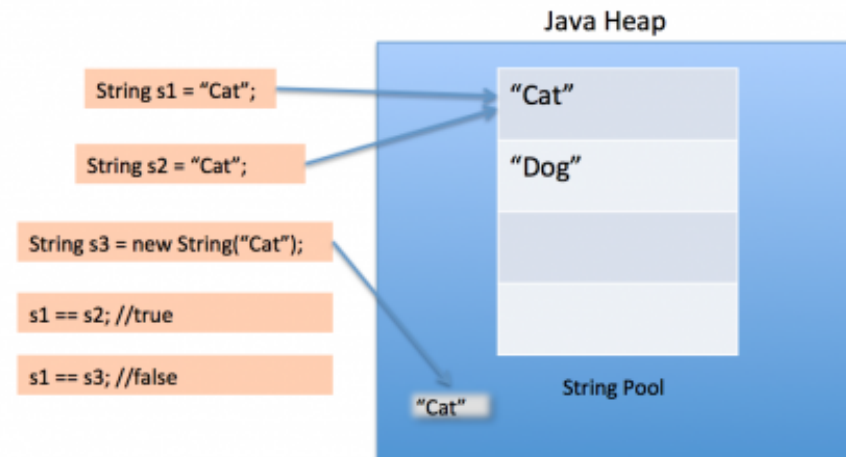
String - StringBuilder - StringBuffer- Mutable- Immutable



- Strings are object representation of character sequences
- **Strings** are **IMMUTABLE** object. The original object value cannot be changed. String variables are stored in “constant string pool”.
 - There is two ways to make a String object: using new keyword or by giving the literal value
 - Literal Strings will be stored in the String pool
- **MUTABLE Strings** made with **new** keyword will be stored in the **heap memory** like other objects
- **StringBuilder** and **StringBuffer** are **MUTABLE** String objects. If the values are changed then the new value replaces the older value.
- **StringBuffer** is **synchronized** and therefore it is slower than StringBuilder.

STRING METHODS

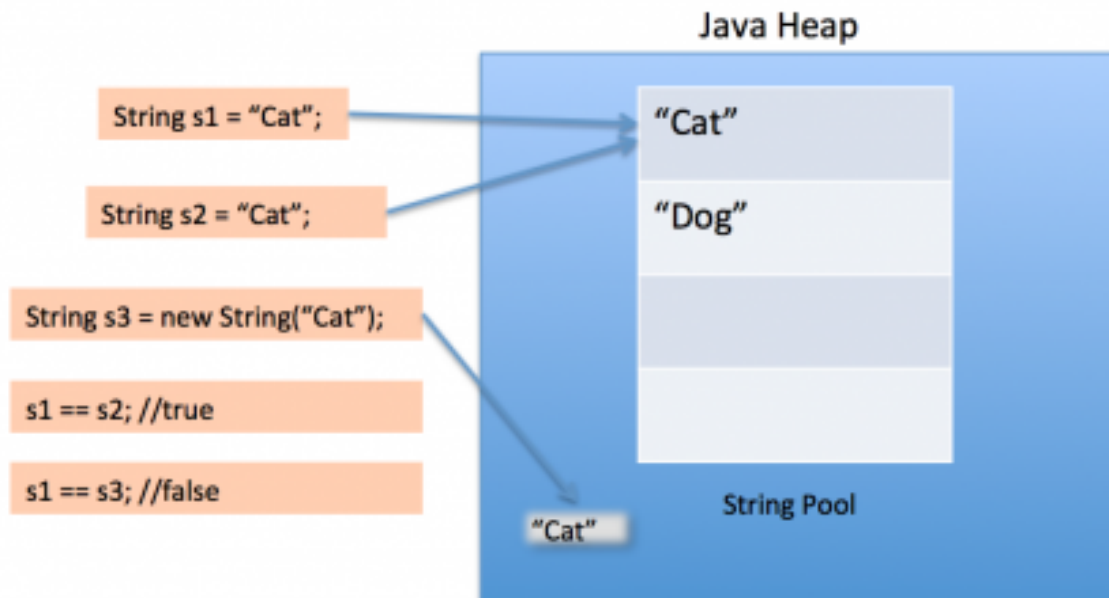
equals()	length()	charAt()	contains()
indexOf()	toLowerCase()	toUpperCase()	replace()
substring()	trim()	startsWith()	endsWith()



String Pool and Heap in Java



- String Pool in java is a pool of Strings stored in [Java Heap Memory](#). We know that String is a special class in java and we can create String objects using a new operator as well as providing values in double-quotes.
- String Pool is possible only because [String is immutable in Java](#) and its implementation of [String interning](#) concept. String pool helps in saving a lot of space for Java Runtime although it takes more time to create the String.
- When we use double quotes to create a String, it first looks for String with the same value in the String pool, *if* found it just returns the reference *else* it creates a new String in the pool and then returns the reference.



```
package com.journaldev.util;

public class StringPool {

    /**
     * Java String Pool example
     * @param args
     */
    public static void main(String[] args) {
        String s1 = "Cat";
        String s2 = "Cat";
        String s3 = new String("Cat");

        System.out.println("s1 == s2 :"+(s1==s2));
        System.out.println("s1 == s3 :"+(s1==s3));
    }
}
```

Output of the above program is:

```
s1 == s2 :true
s1 == s3 :false
```



- Classes act as a blueprint of objects, which are real world representations of objects, in programming
- Each object is referred to as an instance of the class it is created from.
- Object is a member or instance of a class
- Class is declared using class keyword, Object is created through new keyword mainly.
- Everything in java is written in a .java file which will be compiled to a .class file
 - Note: The java file and class file should have the same name
- Objects are created by using new keyword in the following format:
 - `ClassName referenceName = new ClassName ()`

this & Super Keywords [this. - this() - Super. – Super()]



- **this** is used to call the current class members (variables and methods).
 - this.variableName in the sameClass
 - this.methodName in the sameClass
 - this can be used anywhere in a class, except static block or static method
- **this()** is used to call a constructor in the same/current class
 - If this() is called in an overloaded constructor, it must be the first line
- **Super** is used to call the parent class members (variables and methods)
 - Super.variableName from the SuperClass
 - Super.methodName from the SuperClass
 - Super. can be used anywhere in a class, except static block or static method
- **Super()** is used to call the constructor from the parent class
 - **Super()** must be the first statement in the constructor.

Class members – METHODS



- Methods are functions which perform certain actions
- There is two types of methods: void and return
 - void: Methods that perform some action without returning any value
 - return: Methods that will perform an action and return a single value or object back
- Methods can be invoked by using an object following by the dot operator and method name with parentheses at the end

```
public class Person {  
    String name;           // instance variables  
    int age;  
  
    public void printAge(){ // This is a void method named printAge.  
        System.out.println(age); // It will print out the object's age instance  
    }                          // value.  
  
    public int getAge(){    // This is a return type method named getAge.  
        return age;        // It will return the instance age value of  
    }                      // the object acted on as an int  
  
    // The return keyword is how these method give a value  
  
    public static void main(String[] args) {  
        Person james = new Person(); // Object of Person class is created james reference  
        james.age = 20;  
        james.name = "james";  
  
        james.printAge(); // The void method is invoked and the age of  
        // the object (20) will be printed to the console  
  
        int ageVal = james.getAge(); // The return method is invoked and the age instance  
        System.out.println(ageVal); // variable value will be returned so it must be caught  
        // in order to be used. Then the variable ageVal will  
        // print the value that was returned from the method  
    }  
}
```

Continue

Class members – METHODS



- Methods can have parameters which act as required information to execute a method
- Any parameters declared in a method act as local variables for that method
- Multiple parameters can be given by separating each variable with a comma

```
public class Person {  
  
    int age;  
  
    void setAge(int age) { // the set age method will  
        this.age = age;   // accept an int value and  
    }                     // store that value to the  
                           // age instance variable  
  
}
```

STATIC Keyword (Static variable, Static Method)



Static means: you can access them without creating an object, just by using a classname.

- Static members belong to class itself, they are also called class members.
- We can call static members by using class name or object
- But Static methods can not call Non Static members
- **What can be static?** Variables, method, block, inner class can be Static.

static variables -> is also called <shared variable>, it will have only one copy of it. Every object of the class will share the value of it. It is called by ClassName.

static methods -> a method that can be called by classname, without creating an object

- > objects can also call static methods.
- > Only static variables can be in a static method
- > static methods can only access other static variables or methods.
- > static methods cannot call/use instance variables or non-static methods. But non-static method can call a

static method.

static blocks -> code in static block executes only ONCE and BEFORE EVERYTHING ELSE, whenever the class is used.

static inner/nested classes

```
import static java.lang.Math.*; // imports all static members
                                // of the Math class

public class House {

    static int numberOfHouses; // static int variable

    static int getHouse() {      // static return type method
        return max(10,20);      // uses static max method from
                                // Math class
    }

    static {                    // static block ran and getHouse
        numberOfHouses = getHouse(); // method is called which
                                // stores a number into the static
                                // variable numberOfHouses
    }

    public static void main(String[] args) {
        System.out.println(House.numberOfHouses);
    }
    // The value of numberOfHouses is printed after
    // the static block and getHouse method are
    // finished executing
}
```



Local, Instance and Static variables

- **Local variable** is typically used inside a method, constructor, or a **block** and has only local scope. The best benefit of having a local variable is that other methods in the class won't be even aware of that variable.
- **Instance variable** is a variable which is declared within a **class**, but outside a method. Instance Variable belongs to the OBJECT. Can be called by object name. Every object of that class will create it's own copy of the variable while using it. Thus, any changes made to the variable won't reflect in any other instances of that class and will be bound to that particular instance only.

Numbers default to 0, Objects default to null, Booleans default to false

- **Static (class) variable** belongs to the CLASS, can be called through class name. Static variables are

declared with the static keyword in a class, but outside a method. Static variables are also called shared variable, it will have only one copy of it. Every object of the class will share the value of it.

```
• public class VariableExample{
•   int myVariable; //instance variable
•   static int data = 30; //static variable
•
•   public static void main(String args[]){ //main method
•       int a = 100; //local variable
•       VariableExample obj = new VariableExample(); //declare object
•
•       System.out.println("Value of instance variable myVariable: "+obj.myVariable); //instance
        variables can be called by object name
•       System.out.println("Value of static variable data: "+VariableExample.data); //instance
        variables can be called by className
•       System.out.println("Value of local variable a: "+a);
•   }
• }
```

Output:

- Value of instance variable myVariable: 0
- Value of static variable data: 30
- Value of local variable a: 100

Class members – INSTANCE VARIABLES



- Instance variables are the data items inside of classes but outside of a method
 - These variables have datatypes as any other variable
- Every instance, which refers to each object, created from a class will have those instance variables
- These variables have default values
 - Numbers default to 0, Objects default to null, Booleans default to false
- Objects can refer to their instance variables with the dot “.” operator

```
public class Person {  
  
    String name;    // Instance variable of String called name  
    int age;        // Instance variable of int called age  
  
    public static void main(String[] args) {  
        Person joe = new Person(); // Created a Person object called joe  
        joe.name = "Joe";           // assigned "Joe" to the name instance variable  
        joe.age = 20;               // assigned 20 to the age instance variable  
  
        System.out.println(joe.name); // prints the value of name from object joe  
        System.out.println(joe.age);  // prints the value of age from object joe  
    }  
}
```

OVERLOADING Method



- Overloading mean having multiple methods with the same method name but different parameters in the same class. This allows us to have more than one method with same name.
- Method overloading is achieved by having **method with the same name that have a different number of parameters or have a different number of parameters**. In short, the method signature must be different to overload a method
- It doesn't matter, **Return type** can be **same or different** in method overloading.
- It doesn't matter, **Access Modifier** can be **same or different** in method overloading.
- When a method is overloaded the arguments will match to the best method that matches those parameter types and execute that method. Methods cannot be ambiguous with these parameters

Example: sort method of Arrays class

```
Arrays.sort(int[] arr)
```

```
Arrays.sort(String[] arr)
```

Main method can be overloaded if we pass different parameters.

Method overloading improves the reusability and readability. It's easy to remember (one method name instead of remembering multiple method names)

Overloading vs Overriding



Method Overloading	Method Overriding
Same method name with different parameters	Same method name with same parameters
Method overloading is achieved in the same class	Method Overriding occurs in two classes that have IS-A relationship
Return type can be same or different	Return type must be same or covariant type
Access modifier can be same or different	Access modifier must be same or more visible
Private and final methods can be overloaded	Private and final methods CANNOT be overridden

Constructor



- Constructors are the special methods without any return type. Constructors have the same name with the className
- Constructors are special methods, which are called whenever the new keyword is used to create an object of a class
- By default every class always has a default constructor with no parameters. This default constructor is no longer there if a constructor is created manually
- Constructors usually act to initialize instance variables or perform actions that need to be taken whenever an object of a class is created.
- Like methods, constructors can also be overloaded. The same rules apply. This is done by providing a constructor with with a different parameters
- this() can be used to call the overloaded constructors with other parameters

```
public class Animal {  
  
    String species;  
    int size;  
    double height;  
  
    public Animal(String species) {  
        this.species = species;  
    }  
  
    public Animal(String species, int size) {  
        this(species);  
        this.size = size;  
    }  
  
    public Animal(String species, int size, double height) {  
        this(species, size);  
        this.height = height;  
    }  
  
}
```

Constructor (detail to know. Do not talk about this)



In this example the first constructor will be invoked
Because the constructor call does not have parameters,
but as Shown it is possible to also overload constructors

House -> Class name

houseOne -> reference name of object

new House() -> used to invoke constructor

At the Interview
Do not read this page. This is detail to know about the constructor

```
public class House {  
  
    String address;  
  
    // constructor with no parameters  
  
    public House () {  
        this.address = "No address yet";  
    }  
  
    // overloaded constructor that accepts  
    // an address and stores to instance  
  
    public House(String address) {  
        this.address = address;  
    }  
  
    public static void main(String[] args) {  
        // In the line below the House class  
        // constructor is called and an object  
        // of the class is created  
        House houseOne = new House();  
    }  
}
```

Constructor – Rules for Constructor chaining



1. Any constructor call must be call from another constructor
2. In order to call an overloaded constructor, MUST use this() keyword
this(): can only be used in a constructor calling another constructor (DO NOT USE NAME OF CONSTRUCTOR)
3. Constructor call MUST be at FIRST line of a constructor. If we call at later step ==> Compiler error
4. A Constructor can call ONLY ONE Constructor. If we call twice ==> Compiler error
5. A Constructor can NOT call ITSELF. If it calls itself ==> Compiler error
6. A Constructor can NOT contain itself ==> Meaning if a constructor called another, the latter cannot call the first (because of rule 5)

At the Interview

Do not read this page. This is detail to know about the constructor

Constructor vs Method?



- **Constructor** is a special method in the class that is executed whenever object is created. Doesn't have a return type and constructor's name must be same as the class name.
- Constructor is called automatically when a new object is created.
- Constructor is invoked implicitly.
- The Java compiler provides a default constructor if we don't have any constructor.
- Constructors are not inherited by child classes
- **Method:** Collection of statements that are grouped together to perform an operation.
- Method have a return and the method's name may or not be same as the class name.
- Method is invoked explicitly.
- Method is not provided by compiler in any case. Methods are inherited by child classes.



Final vs Finalize vs Finally?

- **final** is a keyword and used to apply restrictions on class, method and variable.
 - final is a keyword
 - final class cannot be inherited
 - final method cannot be overridden
 - final variable value cannot be changed.
- **finally** is a block and used to place important code, it will be executed whether exception is handled or not.
 - finally is a block which is used for exception handling along with try and catch blocks
- **finalize** is a method and used to perform clean up processing before object is garbage collected.
 - finalize() method is protected method of java.lang.object class
 - it is inherited to every class you create in java
 - finalize() method is used to perform some clean up operations on an object before it is removed from memory by the garbage collector method. [gc.garbage()]

Access Modifier



- Access modifiers can be applied to variables, classes, methods, constructors

Public – visible to the the whole project

Private – visible only within the same class

Protected – visible only in the same package or sub classes in other packages (inheritance)

Default – visible only in the same package. If no modifier is given this is the default modifier

Public > Protected > Default > Private

equals() vs == in Java



- Equals() method is used to compare the values of the objects.
- “==” or equality operator in Java is a binary operator provided by Java programming language and used to compare primitives and objects. It compare the references and also values of the objects.

Java pass by value or pass by reference?



- - Java is a “pass-by-value” language. This means that a copy of the variable is made and the method receives that copy. Assignments made in the method do not affect the caller.
- **Passing by value** means that the value of the function parameter is copied into another location of your memory, and when accessing or modifying the variable within your function, only the copy is accessed/modified and the original value is left untouched. Passing by value is how your values are passed on most of the time.
- **Passing by reference** means that the memory address of the variable (a pointer to the memory location) is passed to the function. This is unlike passing by value, where the value of a variable is passed on. In the examples, the memory address of myAge is 106. When passing myAge to the function increaseAgeByRef, the variable used within the function (age in this example) still points to the same memory address as the original variable myAge (Hint: the & symbol in front of the function parameter is used in many programming languages to get the reference/pointer of a variable).

Break vs Continue



- break leaves a loop, continue jumps to the next iteration.

Java 7 vs Java 8



JAVA 7	JAVA 8
String in Switch statement	Lambda Expression
Multiple Exceptions Handling	Pipelines and Streams
	date and time in API
	Java 8 interface changes
	Static Method, Default Method which are used in Interface too

Lambda Expression in Java



- Lambda Expressions were added in Java 8. A lambda expression is a short block of code which takes in parameters and returns a value. Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method.

- Example: Use a lambda expression in the ArrayList's forEach() method to print every item in the list:

```
import java.util.ArrayList;

public class MyClass {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<Integer>();
        numbers.add(5);
        numbers.add(9);
        numbers.add(8);
        numbers.add(1);
        numbers.forEach( (n) -> { System.out.println(n); } );
    }
}
```

- Example: Use Java's Consumer interface to store a lambda expression in a variable:

```
import java.util.ArrayList;
import java.util.function.Consumer;

public class MyClass {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<Integer>();
        numbers.add(5);
        numbers.add(9);
        numbers.add(8);
        numbers.add(1);
        Consumer<Integer> method = (n) -> { System.out.println(n); };
        numbers.forEach( method );
    }
}
```

What is Thread-safe or Synchronized?



- Thread safety is very important, and it is the process to make our program safe to use in multi-threaded environment, there are different ways through which we can make our program thread safe.
- **Synchronization** is the easiest and most widely used tool for thread safety.
- JVM guarantees that synchronized code will be executed by only one thread at a time.
- JAVA keyword synchronized is used to create synchronized code and internally it uses locks on Object or Class to make sure only one thread is executing the synchronized code.
- I mean Java synchronization works on locking and unlocking of the resource, so no thread enters into synchronized code.
- We can use synchronized keyword in two ways, one is to make a complete method synchronized and other way is to create synchronized block.

Multithreading in Java



Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread.

So, threads are light-weight processes within a process. Threads can be created by using two mechanisms :

1. Extending the Thread class
2. Implementing the Runnable Interface

Detail:

1. In order to Thread creation by **extending the Thread class**: We create a class that extends the `java.lang.Thread` class. This class overrides the `run()` method available in the Thread class. A thread begins its life inside `run()` method. We create an object of our new class and call `start()` method to start the execution of a thread. `start()` invokes the `run()` method on the Thread object.

2. In Order to Thread creation by **implementing the Runnable Interface**: We create a new class which implements `java.lang.Runnable` interface and override `run()` method. Then we instantiate a Thread object and call `start()` method on this object.

OOP PRINCIPLES



- OOP Concept
- ENCAPCULATION
 - ENCAPCULATION in my project
- INHERITANCE
 - Singel, Multi Level and Hierarchical INHERITANCE
 - IS- A & HAS-A (Composition) Relationship
 - CONSTRUCTOR and Inheritance
 - STATIC MEMBERS and INHERITANCE
 - OVERRIDING Methods
 - OVERRIDING Methods Rules
 - Super keyword
 - THIS vs SUPER
 - THIS() vs SUPER()
 - INHERITANCE in my project
- ABSTRACTION
 - ABSTRACT CLASS
 - INTERFACE
 - ABSTRACTION in my Project
 - Differences: Abastract Class VS Interface
- POLYMORPHISM
 - Static – Dynamic Polymorphism
 - REFERENCE CASTING – Polymorphism
- Differences: Abastract Class VS Interface
- Differences:
- Differences:
- Differences:
- Differences:



- **Encapsulation:**

We can hide direct access to data by using private key and we can access private data by using getter and setter method.

- **Inheritance:**

It is used to define the relationship between two classes. When a child class acquires all properties and behaviors of parent class known as inheritance. Child class can reuse all the codes written in parent class. It provides the code reusability.

- **Abstraction:**

It is a process of hiding implementation details and showing only functionality to the user. Abstraction lets you focus on what the object does instead of how it does it.

- **Polymorphism:**

It is an ability of object to behave in multiple form. The most common use of polymorphism is Java, when a parent class reference type of variable is used to refer to a child class object.

E.g.: `WebDriver driver = new ChromeDriver();`

We use method overloading and overriding to achieve Polymorphism.



- Encapsulation allows the programmer to hide and restrict access to data. More specifically encapsulation will allow an object to hide its variables and methods from being accessed outside of a class. *To achieve encapsulation:*

- 1) Declare the variables with private access modifier
- 2) Create public getters and setters that allow access to those variables

- We can provide only getter in a class to make the class immutable. (Read only)
- We can provide only setter in a class to make the class attribute write-only.

```
public class Person {  
  
    private String name;  
    private int age;  
  
    public int getAge() {  
        return age;  
    }  
  
    public void setAge(int age) {  
        this.age = age;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

```
public static void main(String[] args) {  
  
    Person p1 = new Person();  
  
    p1.setName("Mike");  
    p1.setAge(27);  
  
    System.out.println("Name: " + p1.getName() +  
                        "\nAge: " + p1.getAge());  
  
}
```


ENCAPCULATION in my project



In my project I created multiple POJO/BEAN classes in order to manage test data and actual data. I take JSON from API response and convert to object of my POJO class. All the variables are private with getters and setter. Also in order to store credentials or sensitive data in my framework I have use encapsulation, configuration reader also known as property file or excel sheet to hide data from the outside. I use Apache POI if the data store in Excel in order to extract/read and modify data.



- Inheritance allows one class to inherit content from another class (fields and methods), or in other words, allows a class to be a copy of another class. It allows code reusability.
- Super class (also known as parent or base class): is the class where the fields and methods that are being inherited or passed on
- Sub class (also known as the child or derived class): is the class getting the inherited fields and methods
- In order to use inheritance in java we use extends keyword
- One class cannot extend more than one super class. Every class in java inherits Object class

What is inherited in Java:

- All public and protected variables and methods
- All default variables and methods if the super and sub classes are in the same package

What is NOT inherited in Java:

- Constructors are not inherited
- Any **private** variables or methods are not inherited

Singel, Multi Level and Hierarchical INHERITANCE



```
public class A{  
  
}  
  
public class B extends A{  
  
}
```

Single inheritance

```
public class A{  
  
}  
  
public class B extends A{  
  
}  
  
public class C extends B{  
  
}
```

Multi-level inheritance

```
public class A{  
  
}  
  
public class B extends A{  
  
}  
  
public class C extends A{  
  
}  
  
public class D extends A{  
  
}
```

Hierarchical inheritance

IS- A & HAS-A (Composition) Relationship



- IS-A relationship can be used to describe the relationship between classes that are using inheritance

IS-A:

- German Shepherd IS-A Dog
- Dog IS-A Animal
- German Shepherd IS-A Animal

HAS-A:

- Car HAS-A engine
- Dog HAS-A tail



- The super class constructor always executed before the sub class constructor
- It is possible to call the super class' constructor using keyword Super()
- The default no parameter constructor automatically invokes its super class constructor
- Rules:
 - If this super constructor is called explicitly it must be the first line in the sub class constructor: super()
 - The parameters must match with parent constructor. If parent class has only constructors with parameters, then an explicit call must be made matching those parameters

STATIC MEMBERS and INHERITANCE



- If the access modifier allows inheritance, static members will also be inherited
- **Static variables** will be shared for all objects related to that class. All the object from the parent to the child
- **Static methods** can be called using the parent class name or sub class name



- After inheriting a method from the parent class, the child class can change the action. So we are able to keep the same method, but change the implementation to better match the child class' needs. In method Overriding;

1. There **must be is-a relationship (inheritance)** between the classes.
2. The **method** must have the **same name AND same parameters** as in the parent class
3. **Access modifier** needs to be same or more visible
4. **Return type must be same** or **covariant type** (same class type or sub class type)

- We can use the **@Override annotation** before the method to tell the compiler we are overriding a method. It will help ensure the method is overridden correctly.

- Ex: toString() method

- Unlike methods, it is not possible to override a variable, but it is possible to hide them. Variable hiding occurs when a variable is declared with the same name as a variable from the super class

OVERRIDING Methods RULES



1. There must be is-a relationship (inheritance)
2. The method must have the same name AND same parameters as in the parent class
3. Access modifier: Needs to be same or more visible
4. Return type:
 - must be same or
 - covariant type (same class type or sub class type)

***Notes:

We cannot override static methods. If you try, they will become hidden

- A hidden method occurs when a child class defines a static method with the same name and signature as a static method defined in a parent class.
- Method hiding is similar but not the same as method overriding.
- The rules for overriding a method are the to hide a method, but in addition, the method must stay static.

Super keyword



The Super keyword in java is a reference variable that is used to refer to parent class objects in inheritance

- Super with variables (Super.variable)
- Super with methods (Super.method)
- Super with constructors (Super())

THIS vs SUPER



- **Super** is used to access/call the parent class members (variables and methods)
- **this** is used to call the current class members (variables and methods).
- Can be used when parameter local variables have the same name as instance variables
- Both can be used anywhere in a class, except static block or static method

THIS() vs SUPER()



- **this()** is used to call an overloaded constructor in the same class.
- **super()** is used to call the constructor from the parent class
- Both **this()** and **super()** must be the first statement in the constructor. This results in one constructor calling another constructor.

INHERITANCE in my project



In my framework I have a TestBase class where I store all my reusable code and methods. My test execution classes, and elements classes will extend the TestBase in order to reuse the code.

My framework follow POM and some pages have similar actions, so I can easily use those similar actions and fields by inheriting some a common class.

What is overriding?

Overriding means changing the implementation of a method that is coming via inheritance from a super class

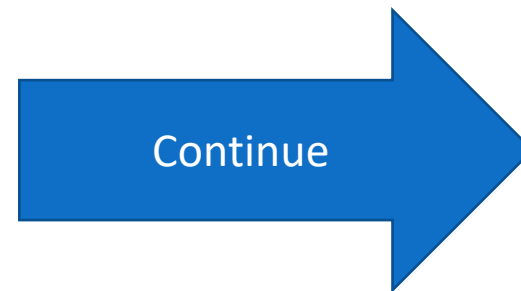
Example: get method

```
WebDriver driver = new ChromeDriver();  
driver.get("URL") ==> opens the url from chrome  
WebDriver driver = new FireFoxDriver();  
driver.get("URL") ==> opens the url from Firefox
```

ABSTRACTION



- Abstraction means IMPLEMENTATION HIDING.
- Abstraction deal with **what** method can do instead of **how** method can do!
- In java, abstraction is achieved via **abstract class** or **interface**
- Abstraction cannot exist without inheritance
- Abstraction is achieved by two way **Abstract Class** and **Interface**



Abstract Class VS Interface



Abstract Class	Interface
Abstract Classes can have abstract and non-abstract methods.	The interface has only abstract methods and with Java-8 static and default methods.
An abstract class can have instance variables.	An interface cannot have instance variables.
An abstract class can have the constructor.	The interface CANNOT have the constructor.
A class can extend one abstract class.	A class can implement multiple interfaces.
The abstract keyword is used to declare an abstract class.	The interface keyword is used to declare an interface.
An abstract class can be extended using extend keyword	An interface class can be implemented using keyword implements
An abstract class can extend another Java class and implement multiple Java interfaces.	An interface can extend another Java interface only.
A Java abstract class can have class members like private, protected, public.	Members of a Java interface are public by default.
Example: <pre>public abstract class Shape{ public abstract void draw(); }</pre>	Example: <pre>public interface Drawable{ void draw(); }</pre>

ABSTRACT CLASS RULES



- To create an abstract class: add keyword `abstract` before the class declaration
- An abstract class cannot be instantiated – An object can never be created from that class
- An abstract class allows creation of abstract methods, which are declaration of method signatures without any implementation. In other words, methods without a body.
- The class which inherits the abstract class is called the concrete class. The concrete class must implement all abstract methods declared from the abstract class
- Variables act following regular inheritance rules

Abstract class called Student →

Abstract method named `attendClass()` →

Concrete class named `LocalStudent` →

Implementation of abstract method

`attendClass` →

```
public abstract class Student{  
    public abstract void attendClass();  
}  
  
public class LocalStudent extends Student{  
    @Override  
    public void attendClass(){  
        System.out.println("Attending in person");  
    }  
}
```

Continue

ABSTRACT CLASS RULES



- Abstract classes can not be instantiated
- Abstract classes may be defined with any number of abstract and non-abstract methods, including none
- Abstract classes can not be private or final. A class can extend only one abstract class.
- An abstract class that extends another abstract class inherits all its abstract methods as its own abstract methods
- The first concrete class that extends an abstract class must provide an implementation for all the inherited abstract methods:

-
- It's possible for an abstract class to inherit another abstract class. In this case implementation of the abstract methods are not required
 - In this case the sub abstract class is not a concrete class, but as soon as a concrete class inherits all the properties of that abstract class it must implement all abstract methods from all super classes

```
public abstract class Student {  
    public abstract void study();  
}  
  
public abstract class LocalStudent extends Student {  
}  
  
public class CollegeStudent extends LocalStudent {  
    @Override  
    public void study(){  
        System.out.print("College student studying");  
    }  
}
```

Continue

ABSTRACT CLASS RULES



- Abstract methods can only be defined in abstract classes or interfaces
- Abstract methods can not be declared private or final (public abstract void dog(); or void dog();)
- Abstract methods can not provide a method body/implementation in the abstract class or interface it is declared
- Implementing an abstract method in a subclass follows the same rules for overriding a method.
- Do abstract classes have a constructor?
 - > Yes, the class is still inheriting the Object class and has a default constructor (but Interface cannot have constructor)
- Can an abstract class have only non-abstract methods?
 - > Yes, an abstract class doesn't need to have any abstract methods.
- If the abstract class doesn't have any abstract methods, can you instantiate?
 - > No, objects of an abstract classes can never be made

ABSTRACT CLASS RULES



- Can you add instance or static variables into abstract class?
 - > Yes (But you cannot add into Interface. Interface can only have constant variable which is static final variable)
- Can you add initializer or static blocks into Abstract class?
 - > Yes
- What access modifiers can abstract methods be?
 - > They can have any access modifier except private
- Can an abstract method be static?
 - > No, abstract methods are meant to be overridden, and only the instance methods can be overridden
- Can a method be abstract and final?
 - > No, we cannot have abstract final methods. Final methods cannot be overridden, and abstract methods must be implemented

INTERFACE



- Interface is the second way to achieve abstraction in java.
- An interface is not a class, but acts similar
- To create an interface the keyword **interface** is used instead of class →
- The main purpose of an interface is providing additional information and behaviors to any class that needs it.
- An Interface also allows creation of abstract methods
- By default any method created in an interface is < public abstract > →
- [Java 8] Only way **to create non-abstract methods** in an interface is to create a default or static method
 - Static methods: use keyword static before the return type of the method. This will allow a method to be created with implementation.
 - Default methods: use keyword default before the return type of a method. This will allow a method to be created with implementation.

```
public interface Teachable{  
  
}
```

```
public interface Teachable {  
  
    public abstract void doHomework();  
    // can be written just:  
    // void doHomework();  
}
```

```
public interface Teachable {  
  
    public static void study(){  
        // implementation  
    }  
}
```

```
public interface Teachable {  
  
    public default void learn(){  
        // implementation  
    }  
}
```

Continue

INTERFACE



- By default any **variable** created will be `< static final >` and must be initialized
- To add an interface to a class, keyword `implements` is used after the class name
- Inheritance allows only parent class, but it is possible to implement multiple interfaces to a class
- This is done by adding a comma after each interface after the class declaration. The class must implement all the abstract methods from all the interfaces
- It's possible to extend interfaces to other interfaces. And unlike regular inheritance interfaces can extend as many interfaces as they wish.

```
interface Moveable {  
    void rotate();  
}  
  
interface Upgradeable {  
    void increaseSize();  
}  
  
public class TV implements Moveable, Upgradeable {  
    public void rotate() {  
        //  
    }  
    public void increaseSize() {  
        //  
    }  
}
```

public static void rotate();

public static void increaseSize();

class TV is the concrete class and it must implement the abstract methods in the interfaces...

Interfaces
can have:

- Constant variables `static final int age = 33;`
- Abstract methods
- Default Methods
- Static Methods

Interfaces
can not
have:

- Constructor
- Blocks
- Instance variables or methods

ABSTRACTION in my Project



- In my framework I have achieve abstraction by using collections or Map, because it's all interface. Most of the cases I come across using List. If we want to access elements frequently by using index, List is a way to go. ArrayList provides faster access if we know index. If we want to store elements and want them to maintain an order, List is a better choice.

- i) `List<String> webs = driver.getWindowHandles();`

- =>create a list first to store web URLs in list

- ii) `findElements` evaluates multiple elements so therefore will assigned to `List <WebElement>`

- iii) To handle dynamic elements store it in the list and identify by index:

- `List<WebElement> all = driver.findElements(By.tagName(""));`

- In my framework I follow POM and had situations where some pages shared similar actions that were similar but worked slightly different, so we usually create an abstract class named `BasePage` to have all common members for every page written in this class such as `getPageTitle()`. Then each Page class (`HomePage`, `LoginPage`, `DashboardPage` etc.) inherit from `BasePage`. Sometimes one may need to change the behavior of methods implemented in superclass. So subclass has freedom to override that method where we use polymorphism. This is how we use Abstract class in real projects.

POLYMORPHISM



- In java Polymorphism is the ability of an object to take many forms. This is done through the reference of an object
- Three reference types an object can have:
 1. Itself
 2. Super classes
 3. Interfaces which are implemented

```
Object obj = new Object();  
[1]   [2]   [3]   [4]
```

[1] reference type
[2] reference name
[1 + 2] reference

[3] new keyword
[4] constructor call
[3 + 4] object

OBJECTS REVIEW

```
public class Animal{  
}  
  
interface Tameable {  
}  
  
public class Dog extends Animal implements Tameable{  
}
```

Possible Dog object references:

```
Dog dog = new Dog();  
Animal dog = new Dog();  
Tameable dog = new Dog();
```

Continue

Static - Dynamic Polymorphism



- The reference of an object will determine which class members the object can access
- If a method is called the object will use the method implementation of the object type. If the method is not overridden in the sub class, the super classes method implementation is executed
- The object will always give the values of the object itself if the reference of the object also has access to that class member. If there is a class member that is out of scope of the reference, then there will be a compile time error.
- Polymorphism is achieved by 2 ways:
 1. *Static polymorphism* is method overloading (Compile time)
 2. *Dynamic polymorphism* is method overriding (Runtime)
- One use of polymorphism is for polymorphic collections. Being able to store different types of objects that share similar reference types into one data structure.

Example: `WebDriver driver = new ChromeDriver();`

=>we are initializing Chrome browser using Selenium WebDriver. It also means we are creating a reference variable (driver) of the interface (WebDriver) and creating an Object (from ChromeDriver class).

Continue



- Reference casting is casting the reference to another reference of the object. Casting does not happen between objects.
- **Down casting:**
Going from a super class reference to sub class reference. This must be done explicitly.
- **Up casting:**
Going from a sub class reference to a super class reference. This is done implicitly.
- The reference type is giving in parentheses next to an object to cast the reference to that reference type.

COLLECTION & MAP & Arrays



- Arrays
- Collection Framework
- Iterable interface
- Collection All in One
- LIST Interface
 - ArrayList - LinkedList – Vector classes
- SET Interface
 - HashSet – LinkedHashSet – TreeSet classes
- QUEUE Interface
 - PriorityQueue, Deque, ArrayDeque – Classes
- MAP Interface
 - HashMap, LinkedHashMap, TreeMap - Classes

- Data structures IN your PROJECT?
- Avoiding ConcurrentModificationException
- Differences: LIST vs SET vs MAP (When...)
- Differences: Array VS ArrayList
- Differences: LIST vs SET
- Differences: ARRAYLIST vs LINKEDLIST
- Differences: ARRAYLIST vs VECTOR
- Differences: ITERATOR vs LISTITERATOR
- Differences: HASHMAP vs HASHTABLE
- Differences: HASHMAP vs TREEMAP
- Differences: HASHSET vs TREESET



- Arrays are a data structure that are used to store multiple values of the given datatype.
- These values are stored as elements that can be accessed through an index number. The index number starts at 0 for the first element.
- Key syntax for arrays is the [] symbols
- The size for arrays is fixed, so it cannot be changed.
- Declaring an array will give the element default values based on the datatype: 0 (integers), null (objects)

Arrays Syntax:

```
datatype[] arrVar = new datatype[arrVar.length()]
```

```
1. int[] nums = new int[size];
```

```
2. Int[] nums = {1,2,3,4,...5};
```

```
int[] list1 = new int[4];  
list1[0] = 11;  
list1[1] = 2;  
list1[2] = 49;  
list1[3] = 5;
```

```
int[] list2 = {76, 3, 45, 4};  
int[] list3 = new int[]{25, 4, 1, 99, 8};
```

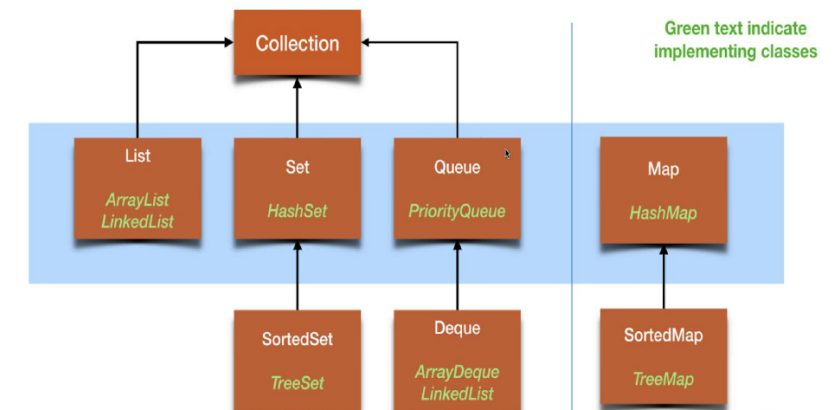
```
int n= list1.length;
```

```
System.out.println("list1 = " + Arrays.toString(list1));  
System.out.println("list2 = " + Arrays.toString(list2));  
System.out.println("list3 = " + Arrays.toString(list3));
```

Collections Framework



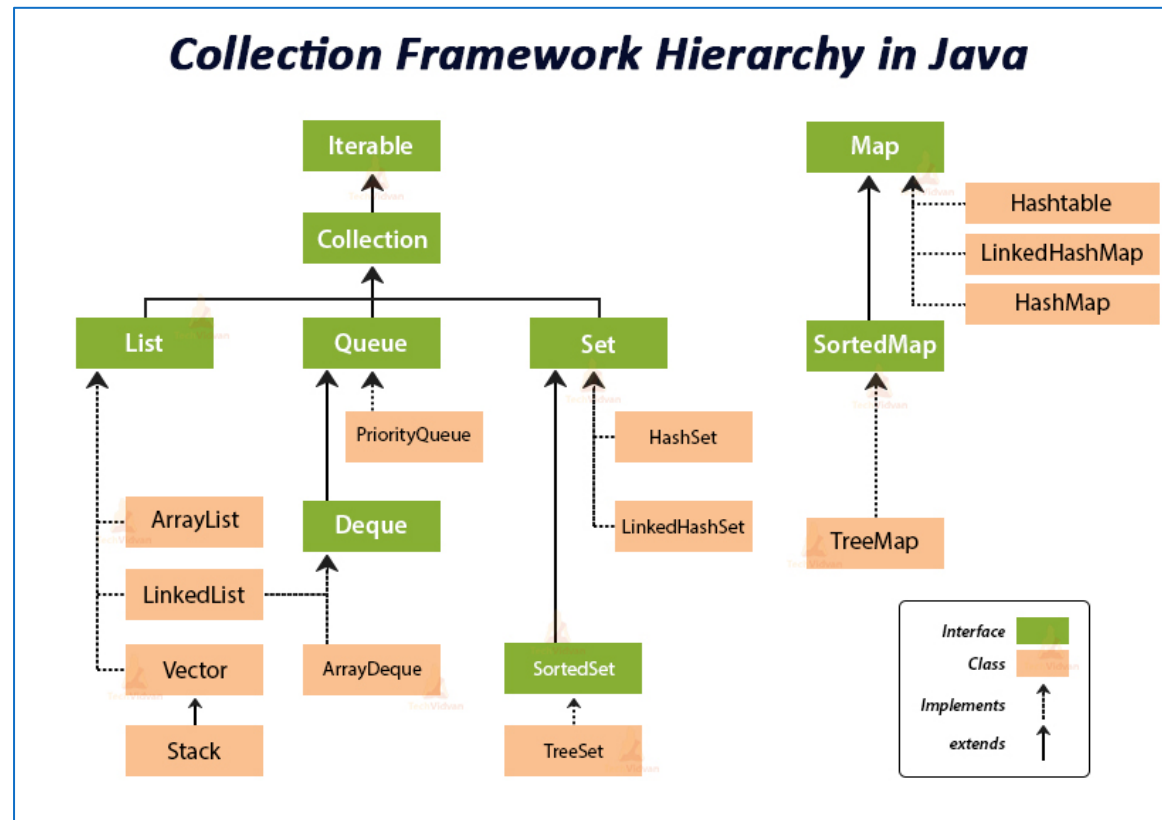
- The collection framework is built up of interfaces and classes that builds data structures with different algorithms to manipulate data.
- The collection in the framework use different implementations to satisfy the algorithms being applied
- Only objects can be stored in these data structures
- At the highest level is the Collection interface, inherits the Iterable interface
- This interface is implemented in all collection classes
- Some method that are declared in Collection interface: add, remove, contains, size, toArray
- The methods declared in the Collection interface are basic operations of the collections that implement them.



Iterable Interface



- This interface is inherited by the Collection interface, so any sub class of the Collection interface also inherits the Iterable interface
- Implementing the Iterable interface allows the objects to be used in the for each loop
- The Iterator interface is an iterator that can be used manually on a collection to enumerate, (continually cycle through the elements if there still is one) except the iterator also allows an element to be removed.





- All collections are Iterable and all collections are core interface. Sync means slow, not sync means fast
- **LIST**: Can store duplicate values, It is ordered. It allows any number of null values. From list we can read a certain value by index. The classes that implement List interface:
 - **ArrayList**(not sync),
 - **LinkedList**(not sync, can work as stack, queue and dequeue) and
 - **Vector**(synchronized, slower)
- **SET**: can only store unique values (no duplicates), and it is not ordered, it is not used with index. The classes that implement Set interface:
 - **HashSet**(not sync, not ordered, allows null),
 - **LinkedHashSet** (no sync, ordered, allow null, and maintains double LinkedList) and
 - **TreeSet**(sorted in ascending order(S-L)) =>(all are class and it implements Set).
 - **SortedSet** (not sync, ordered, comparable interface and it extends Set)
- **MAP** is not collection, it is data structure which is Key and Value format. It cannot have duplicate Key.
 - **HashMap** is not synchronized, allows one null key and more null value. It is faster
 - **HashTable** is synchronized, doesn't allow null key, it is slower.

Collections

LIST (ArrayList, LinkedList, Vector, Stack (extends Vector Class))

ordered, allows duplicate, has an index, accepts multiple null values

- ArrayList: Fast random access; Adding and Removing elements slow; Unsynchronized.
- LinkedList: Slow random access; Adding and Removing elements fast; Unsynchronized.

Methods: .add(), .remove(), .size(), .get(), .set(), .contains(), .isEmpty(), .indexOf(), .addAll(), Collections.sort(), Collections.reverse()

- Vector: Synchronized.

- Stack: subclass of Vector(extends Vector); LIFO principle; Last element's index starts from 1, not from 0.

Methods of Stack: .push(), .pop(), .peek(), .search().

SET (HashSet, LinkedHashSet, TreeSet)

no duplicates, no index, maximum one null.

- HashSet: No order(Random); Accept only one null; No duplicates.
- TreeSet: Ascending order; Can't have null; No duplicates.
- LinkedHashSet: Insertion order; Accept only one null; No duplicates.

Methods: .add(), .clear(), .contains(), .isEmpty(), .remove(), .size(), .toString(), .addAll(), .removeAll(), .toArray(),

MAP (HashMap, TreeMap, Hashtable, LinkedHashMap)

keys and values connected(entry), no index, no duplicate keys, allows null

- HashMap: No order; One null key, multiple null values; No duplicate key, multiple duplicate values.

- TreeMap: Ascending order; No null key, multiple null values.

- Hashtable: No order; No null key, no null value; Synchronized, thread-safe.

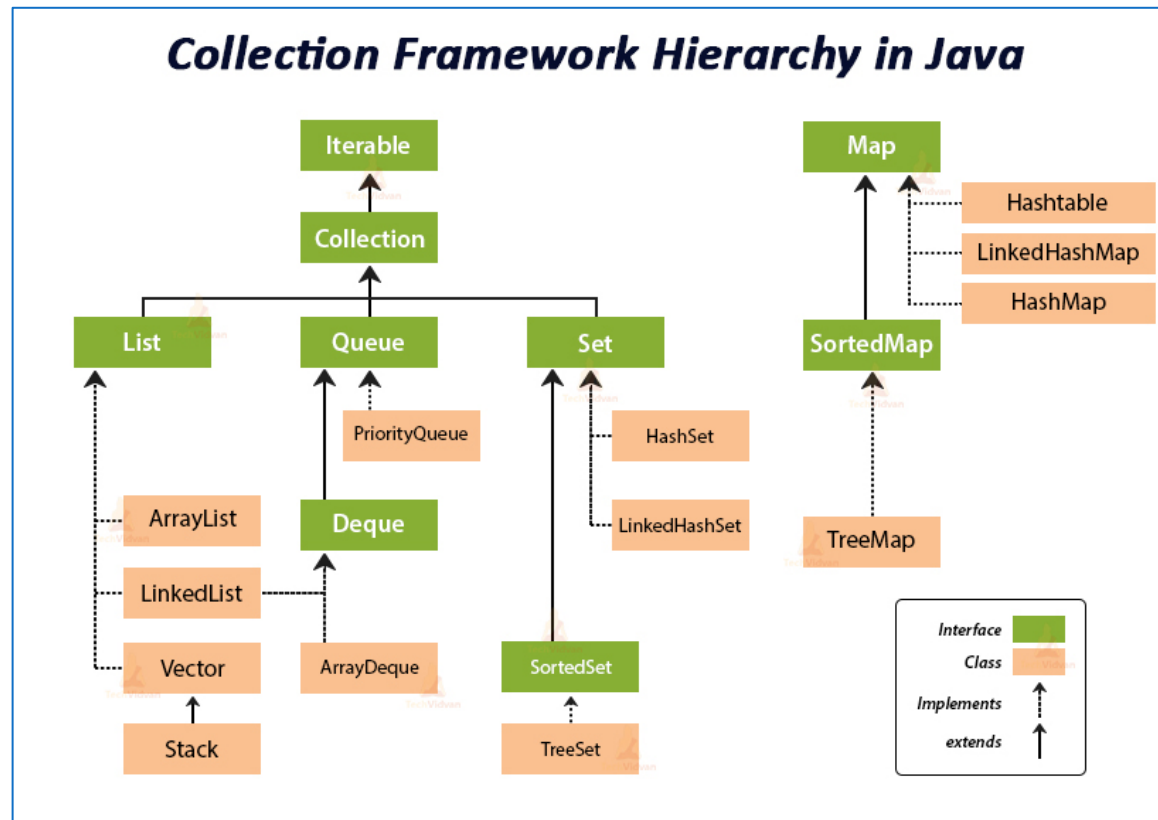
- LinkedHashMap: Insertion order; One null key, multiple null values.

Methods: .put(), .get(), .remove(), .clear(), .containsKey(), .containsValue(), .isEmpty(), .replace(), .size(), .toString(), .values(), .keySet()

List Interface



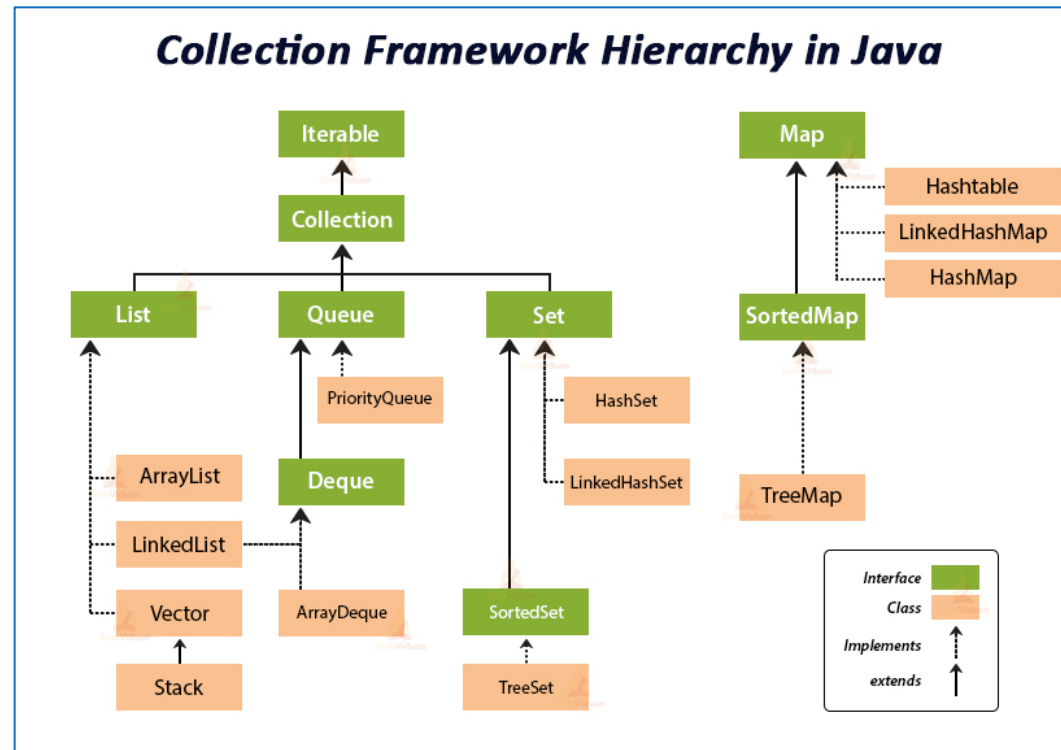
- Inherits the Collection interface
- This is an **ordered** collection, which allows full control of elements that are added. These elements are **accessed by index numbers**, starting from 0.
- This collection **allows duplicate elements**
- Classes that implement List interface: ArrayList, LinkedList, Vector



ArrayList - LinkedList – Vector classes



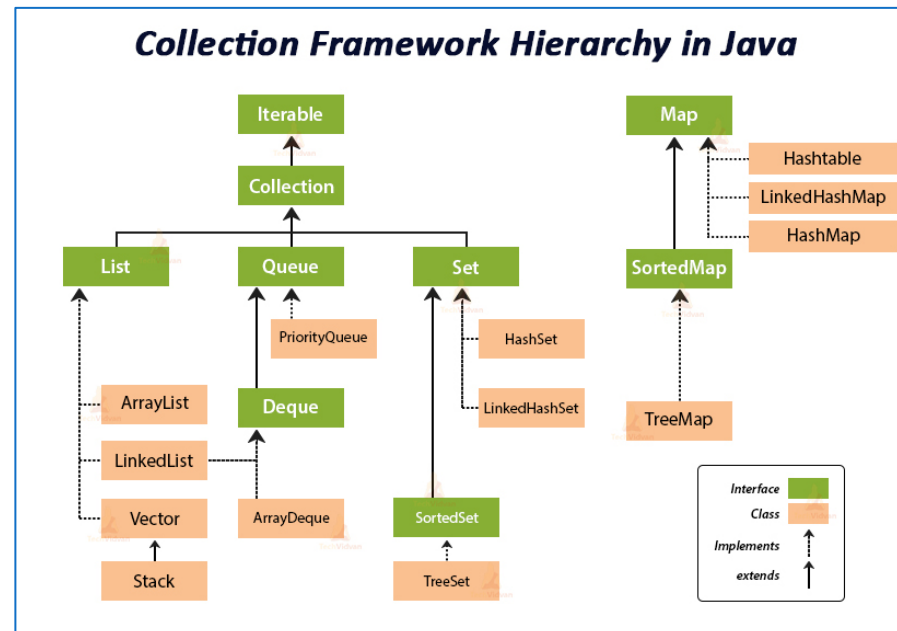
- **ArrayList**: This data structure works similar as an array, but the size is expanded whenever needed
- **LinkedList**: This data structure has nodes, which are objects that have data and reference to another node. These nodes link for the list collection. Also implements Deque interface.
- **Vector**: This data structure is a legacy version of ArrayList. It is a resizable array, but it is synchronized, so it is slower.
- **Stack**: Class inherits the vector class and acts as a LIFO (last in first out) collection



SET Interface



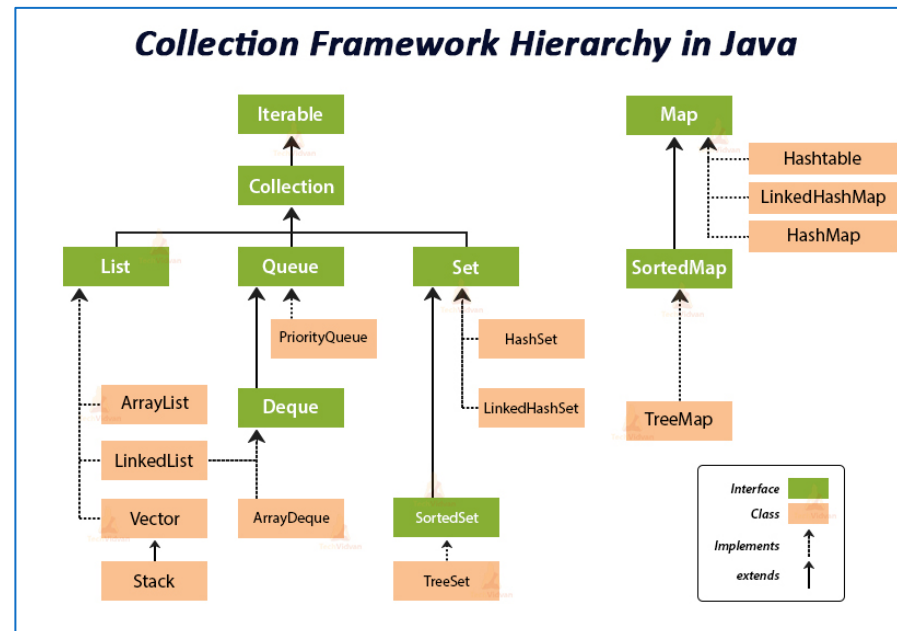
- Inherits the Collection interface
- This collection is an **unordered** data structure, meaning the order elements are added is not maintained
- This collection **does not allow duplicate** elements
- Classes that implement Set interface: HashSet, LinkedHashSet
- **SortedSet** interface inherits the Set interface but **allows the ordering of the elements**. These elements are automatically ordered by their natural order. **TreeSet** implements this interface.



HashSet – LinkedHashSet – TreeSet classes



- **HashSet:** The algorithm used to implement this class is hash table.
 - Short version of hash table algorithm: fast and efficient approach to find elements
- **LinkedHashSet:** Implementing the hash table and linked list this data structure will maintain the insertion order
- **TreeSet:** This data structure is a naturally ordered Set that has additional methods to search for information. Does not allow null element



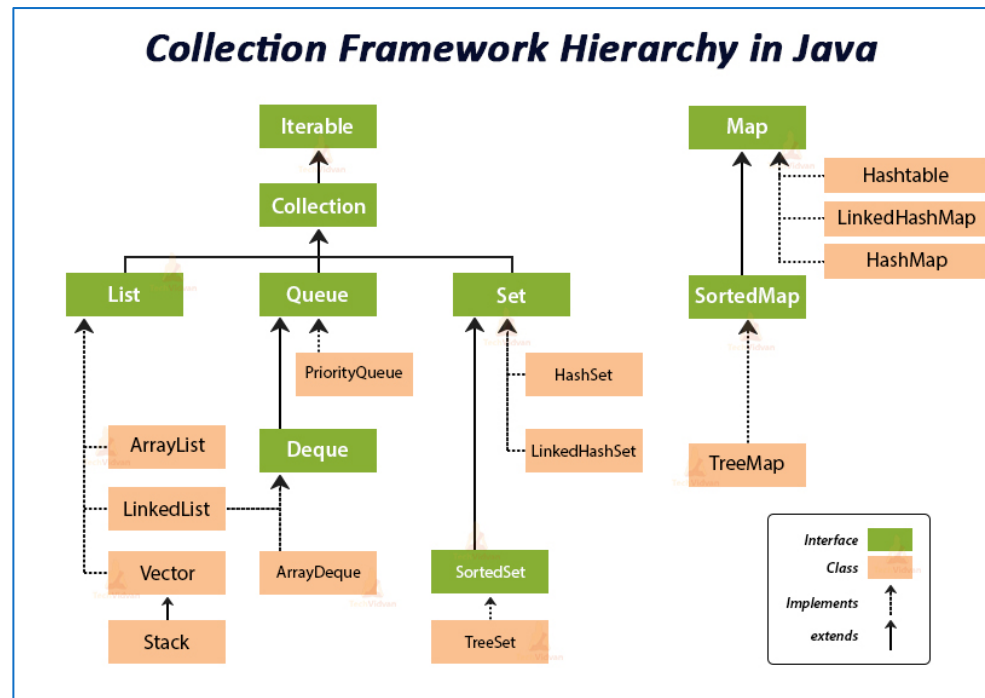
QUEUE Interface



- Inherits the Collection interface
- This data structure has additional features to insert, extract, and look for information. These methods have two forms: one which will throw exception upon failure and one which will return a value

`add() == offer();` `remove() == poll();` `element() == peek();`

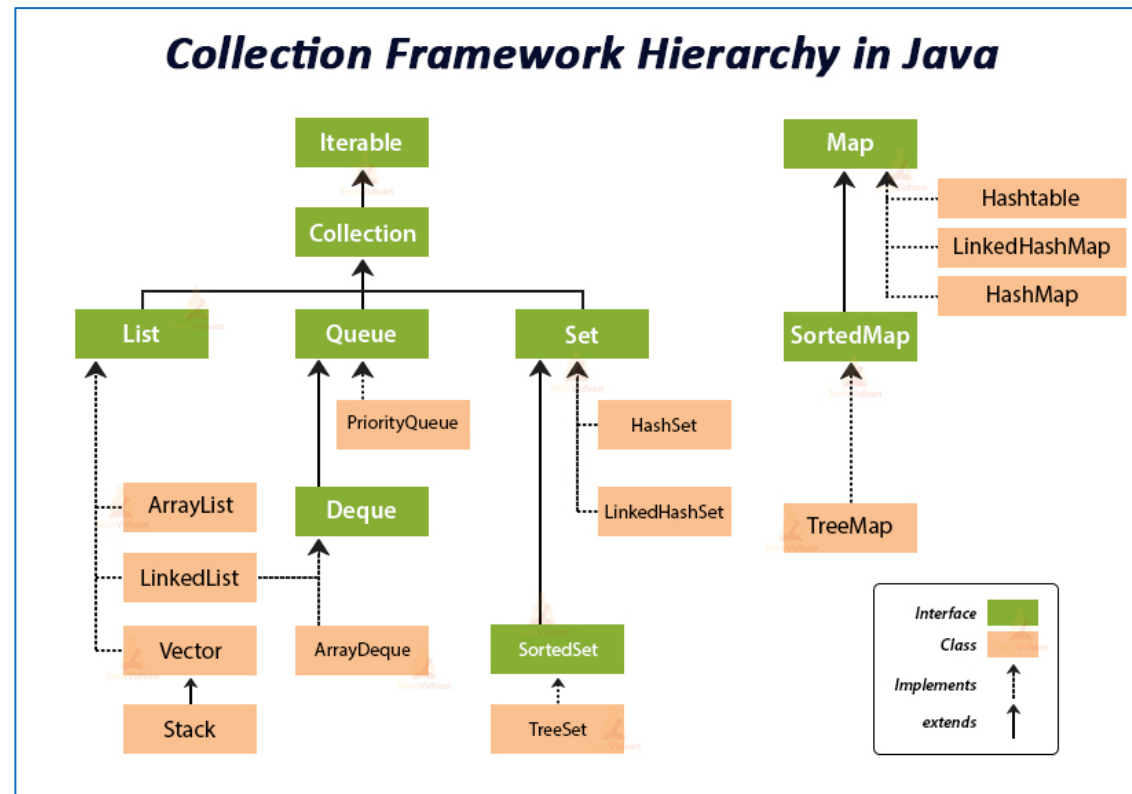
- Queues are usually FIFO (first in first out) meaning elements are added at the tail end of the data structure. BUT this is not always the case. It depends on the classes implementing this interface, they will have to define the insertion direction.



PriorityQueue, Deque, ArrayDeque - Classes



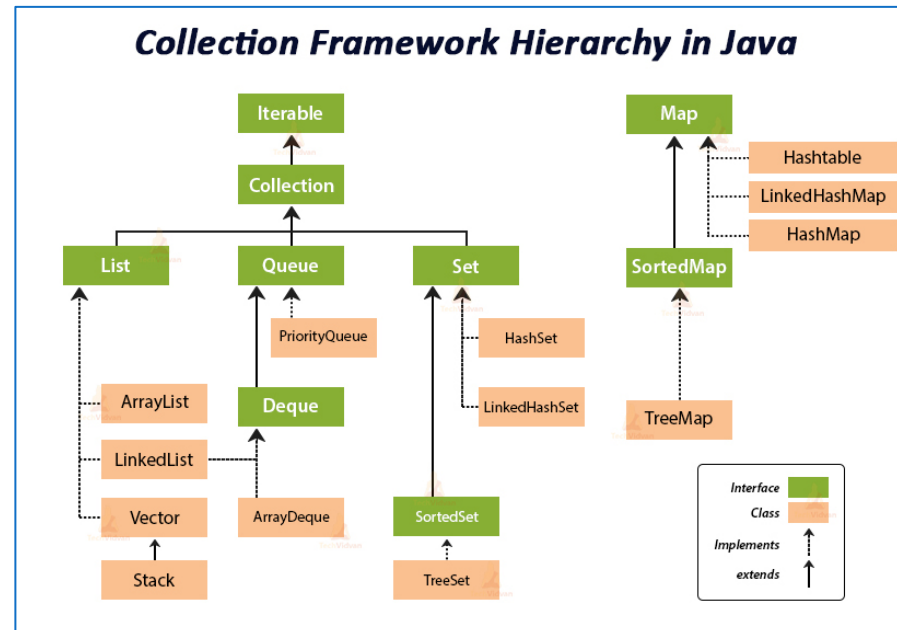
- **PriorityQueue**: This class implements Queue but does not follow **FIFO**. This data structure will store element with their natural order. null is not a valid element
- **Deque**: An interface which allows elements to be used from the beginning or end
- **ArrayDeque**: Class that implements Deque which enables it to access elements from both sides of the data structure. This collection works faster than ArrayList and Stack



MAP Interface



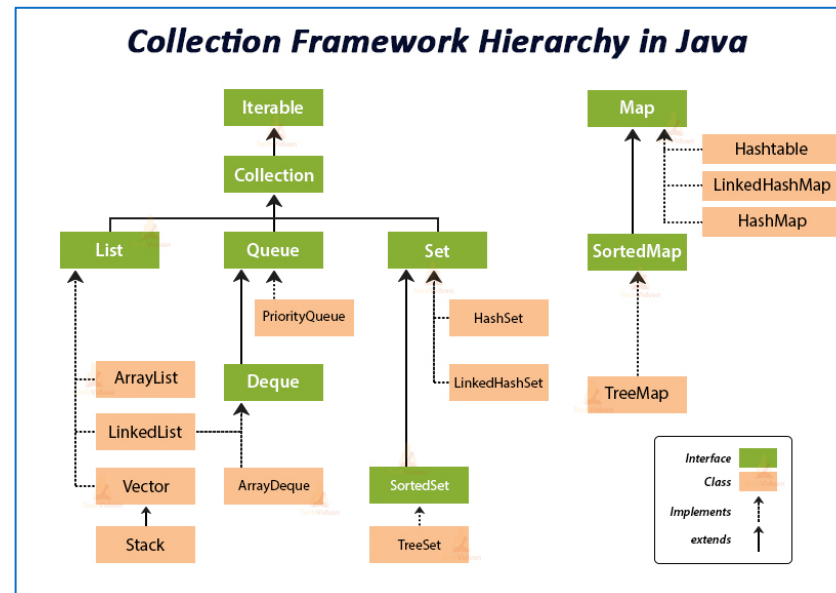
- The map interface does not implement the collection interface and thus is not a part of the collection framework
- A map is a data structure which works with a key/value format. Each key/value pair is called an entry.
- Every key is linked to a single value. Maps do not allow duplicate keys but have no issue with duplicate values.
- Classes that implement the map interface: HashMap, Hashtable and LinkedHashMap
- The Map interface is also implemented in the SortedMap interface, which is implemented to the class TreeMap



HashMap, HashTable, LinkedHashMap, TreeMap - Classes



- **HashMap:** The order of entries is not guaranteed over time. It is not synchronized, allows only one null key, more null value, but no duplicate key. If there is duplicate key, it returns the first one and override others.
- **HashTable:** HashTable is synchronized version of HashMap, so it is slower. Doesn't allow null key. Doesn't allow duplicate key.
- **LinkedHashMap:** This class implements the Map interface and inherits HashMap. The insertion order is maintained.
- **TreeMap:** Implements the SortedMap interface. The data structure will have an ascending order for the keys. Cannot have a null key.



Differences: LIST vs SET vs MAP (When choose...)



Difference between Set, List, and Map

	<u>Duplicate Objects</u>	<u>Order</u>	<u>Null elements</u>
Set	Does not allow.	unordered collection. treeSet(ordered).	allows only one null element
List	Allow.	ordered collection.	allows null elements and can hold many.
Map	Only keys unique.	treeMap(ordered)	one null key/ null values

- If we need to access elements frequently by using index, List is a way to go ArrayList provides faster access with index.
- If we want to store elements and want them to maintain an order, List is an ordered collection and maintains order.
- If we want to create collection of unique elements without duplicates than choose any Set implementation. (HashSet...)
- If we want store data in form Key and Value than Map is the way to go. We can choose from HashMap, Hashtable...

Differences: ARRAY vs ARRAYLIST



Array	ArrayList
Arrays have fixed sizes. Once it is declared, its size does not change.	ArrayLists have dynamic sizes, its size is automatically adjusted
Arrays can hold primitives and object	ArrayList can hold only objects
Arrays can have multidimensional --> [][]	ArrayList can not have multidimensional
Array is a build in data structure	ArrayList is implementing class of List interface in Collection framework

Differences: LIST vs SET



- List vs Set

List > Ordered and Indexed Collection, May contain duplicates

Set > Collection of Unique values, not ordered, no indexing

Data Structure IN your PROJECT



- Do you use data structures in your current automation project?
- Yes, I am very comfortable with data manipulations using Arrays, Collections framework, maps etc. I choose my data structure based on my data and requirements:
 - I have used TreeSet to print dropdown list in for non duplicate values and ascending order
 - I have used HashMaps to compare values from a database with expected values

Differences: ARRAYLIST vs LINKEDLIST



- What is the difference ArrayList vs LinkedList?
 - ArrayList is array based, internally uses array
 - LinkedList is a doubly linked list
 - LinkedList consists of nodes/values that are related to each other
 - ArrayList and LinkedList both maintain ordering
 - ArrayList and LinkedList both allow duplicates
 - ArrayList is better to store and get information. LinkedList is better to manipulate information

Differences: ARRAYLIST vs VECTOR



- Difference between ArrayList vs Vector?
 - ArrayList is not thread safe/not synchronized
 - Vector is thread safe/synchronized
 - ArrayList is faster than Vector
 - Both allow duplicate values and keep ordering
 - Both are implementations of List interface



- Difference between Iterator vs ListIterator:
 - Iterator will cycle through the elements from beginning to end, but ListIterator is able to go in both directions.
 - Iterator can be used in any collection type which implements the collection interface, but ListIterator can only be used for the classes that implement List
 - There is a remove method for Iterator, but ListIterator has more operations available like
 - add()
 - set()
 - remove()

Differences: HASHMAP vs HASHTABLE



- Difference between HashMap and HashTable?
- HashTable is thread-safe/synchronized
HashMap is not thread safe and faster
- HashTable does not allow any null key (will throw NullPointerException)
HashMap allows one null key

Differences: HASHMAP vs TREEMAP



- Difference between HashMap and TreeMap?
- HashMap doesn't maintain any order
TreeMap will store keys in ascending order
- HashMap can have one 'null' key
TreeMap cannot have any 'null' key
- Neither HashMap nor TreeMap are synchronized

Differences: HASHSET vs TREESSET



- Difference between HashSet and TreeSet?
- Both are implementing the Set interface, no duplicate value, no index.
- HashSet performs better than TreeSet
- HashSet is not sorted/ordered, but TreeSet provides natural order (ascending sort).

<code>Set<String> set = new HashSet<>();</code>	store unique value (not duplicate item)
<code>SortedSet<String> sSet = new TreeSet<>();</code>	will remove duplicated data and sort it

How you can avoid ConcurrentModificationException



- How can you avoid the ConcurrentModificationException while iterating through collections?
- You shouldn't change the collection during the iteration, but if you need to, you can use an object such as **ConcurrentHashMap** which is similar to HashMap but will allow you to make modification during iteration.

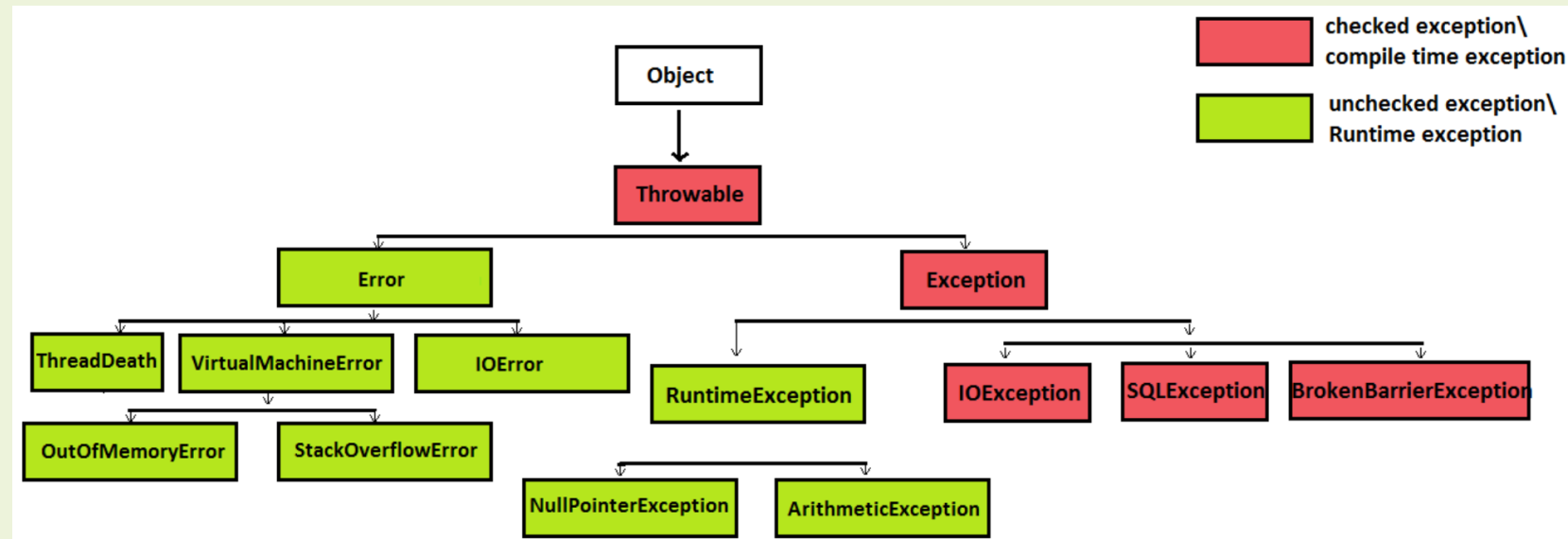
EXCEPTIONS



- [Exceptions](#)
- [How to handle Exceptions in Java](#)
- [Rules of Exceptions](#)
- [Most Common Exceptions in Java and Selenium](#)
- [ElementNotVisible Exception in Selenium](#)
- [Stale Element Reference Exception in Selenium](#)

DIFFERENCES

- [Errors VS Exceptions](#)
- [Compiletime \(Checked\) VS Runtime \(Unchecked\)](#)
- [throw and throws in Java](#)

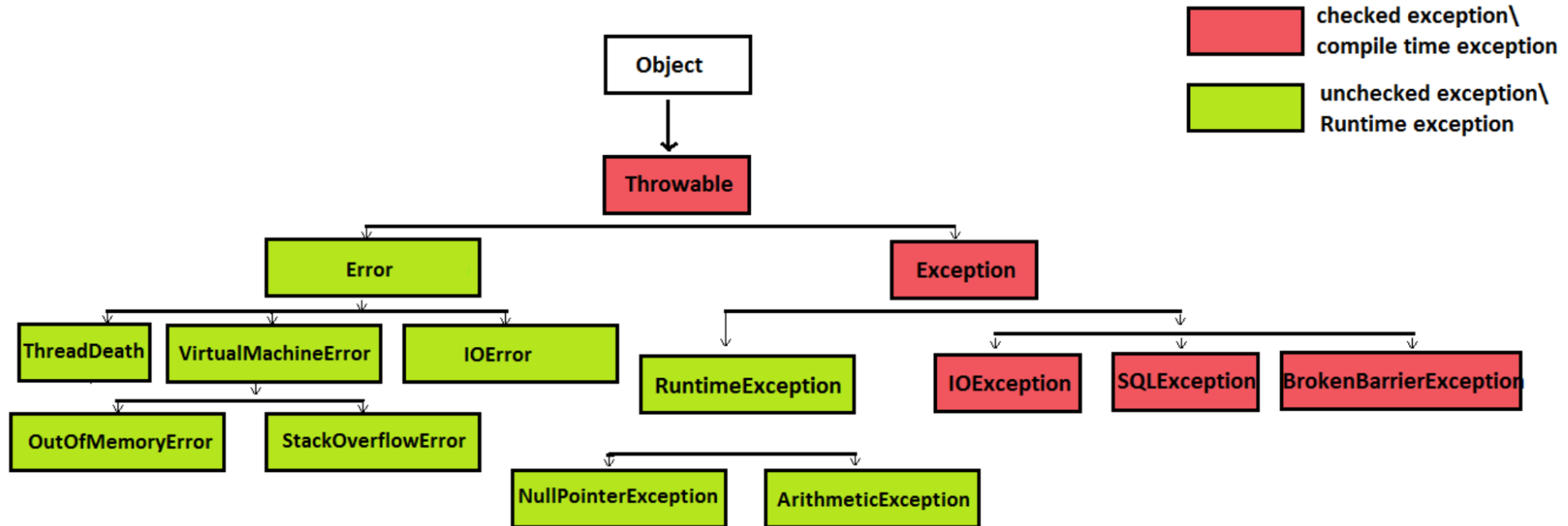


Exceptions



- What is an Exception?

An exception is an unwanted or unexpected event , which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions.



Checked (Compiletime) Exception VS Unchecked (Runtime) Exception



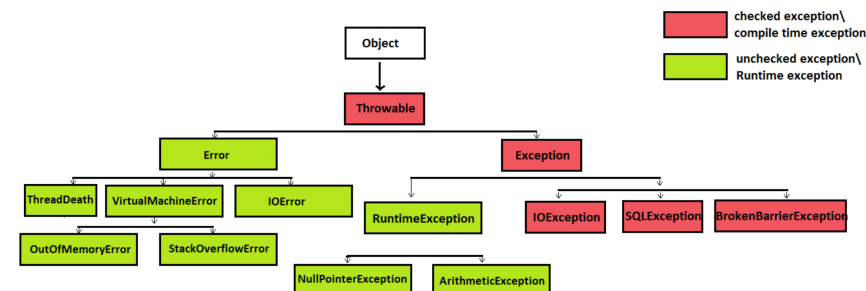
- First I want to remind that Java Exceptions are divided in two categories RuntimeException also known as unchecked Exception and checked (compile time) Exception.
- **1- Checked/Compile time Exception** must be handled before we can run the program. If ignored then program won't compile.
- **2- Unchecked/Runtime Exceptions** where will be thrown during runtime. It compiles without an issue if we don't handle it.
 - It is very challenging to predict. Because it won't complain during compile time.
 - Unchecked exceptions are still can be handled using try/catch block. However if we don't use try catch block it still compiles.

Common Unchecked Exceptions	Common Checked Exceptions
<ul style="list-style-type: none">- NullPointerException- ArrayIndexOutOfBoundsException <p>they are descended from java.lang.RuntimeException.</p>	<ul style="list-style-type: none">- ClassNotFoundException- IOException <p>you need to provide a try catch finally block while performing file operations in Java</p>

Errors VS Exceptions



- Both Error and Exception are derived from Throwable in Java.
- **Error** represent errors which are generally cannot be handled.
 - For examples: *OutOfMemoryError*, *NoClassDefFoundError*
- But, **Exception** represents errors which can be catch and dealt.
 - For examples> *IOException*, *NullPointerException*, *SQL Exception*
- Exception is divided in two categories checked and unchecked Exception. Checked Exception require a mandatory try-catch code block to handle it. Unchecked Exception mostly represent programming errors (*NullPointerException* or *RuntimeException*)
- Errors are unchecked exception and the developer is not required to do anything with these
- All the Errors are Exceptions, but the reverse is not true.
- In general Errors are which NOBODY can control or guess when it happened, on the other hand Exception can be guessed and can be handled



How to handle Exceptions in Java (try&catch – throws – throw)



There are 3 options to handle exceptions: Try&Catch block, Throws keyword and Thrown method **Try&Catch block**. This is real exception handling. The finally block follows a try block or a catch block. Finally block always executes, no matter what happened.

- We use try block for the statement that throws an exception. If Exception is thrown then further execution is terminated in that try block and catch block will start executing statements.

- Try/Catch block avoids program termination.
- We can have multiple catch blocks. Whatever the type of Exception coming from try block will be caught by the matching catch block.

- When creating catch blocks, generic Exception type should be declared last. Otherwise specific exception types will not be reachable.

b. **Throws keyword** which is an Exception declaration in the method signature.

b. **Throw method** which is used to method body.

```
Ex:
String s = "James";
try{
    Thread.sleep(2000);
    System.out.print(s.indexOf(12));
}

Catch block - is used for declaration of exception type and statements that you want to take when exception happens.

catch(InterruptedException e){
    System.out.print("Exception happened.");
}
catch(StringOutOfBoundsException e){
    System.out.print("String is problem");
}
```

```
public void shutdown() throws IOException{
    throw new IOException("Unable to shutdown");
}
```

```
Throw new Exception("is Not able to initialized");
```

throw and throws – Exceptions in Java



- throw and throws are two keywords related to Exception feature of Java programming language.
- throw keyword is used to throw an exception explicitly, on the other hand, throws keyword means passing the exception to next user to handle.
- If we see syntax wise than throw is followed by an instance of Exception class throws is followed by exception class names.
- throw new ArithmeticException ("Arithmetic Exception"); throws ArithmeticException;
- throws is used in method **signature** to declare Exception possibly thrown by any method, for example

```
public void shutdown() throws IOException{  
    throw new IOException("Unable to shutdown");  
}
```

- But throw is actually used to **method body**. `Throw new Exception("is Not able to initialized");`
- In other words; throws keyword cannot be used anywhere exception method signature while throw keyword can be used inside method or static initializer block provided sufficient exception handling.

Rules of Exceptions in Java



1. When using multiple catch blocks, Parent Exception type MUST be used in the end. Otherwise child exception catch blocks will be 'Unreachable code' and it won't be compiled.
2. We use pipe line "|" between exceptions when throwing multiple exceptions.
3. If there are multiple catch block, when exception is caught by first catch block, rest of the catch block will NOT catch any exception.

```
try{  
  
    }catch(SQLException | InvalidKeyException | BadPaddingException |  
    IllegalBlockSizeException | NoSuchAlgorithmException  
    | NoSuchPaddingException e) {  
        e.printStackTrace();  
    }
```




In Java	In Selenium
<p>Unchecked:</p> <ul style="list-style-type: none">• IndexOutOfBoundsException - while working with arrays/strings• NullPointerException - if I forget to instantiate the objects• NumberFormatException,• ClassCastException• ArithmeticException <p>Checked:</p> <ul style="list-style-type: none">• IOException,• SQLException,• FileNotFoundException	<ul style="list-style-type: none">• NoSuchElementException - it throws when locating an element using by findElement. To handle it I declare IMPLICIT WAIT: <pre>driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);</pre>• TimeOutException: To handle it declare EXPLICIT WAIT: <pre>WebDriverWait wait = new WebDriverWait(driver, 5); wait.until(ExpectedConditions.titleIs("text")); wait.until(ExpectedConditions.titleContains("text")); wait.until(ExpectedConditions.elementToBeClickable(WebElement)); wait.until(ExpectedConditions.visibilityOf(WebElement)); wait.until(ExpectedConditions.invisibilityOf(WebElement));</pre>• NoSuchFrameException,• NoSuchAlertException,• NoSuchSessionException, <p>Exceptions when we try to interact with an element (click, sendKeys) :</p> <ul style="list-style-type: none">- StaleElementException- ElementNotVisibleException- ElementNotInteractableException

ElementNotVisible Exception in Selenium



- This exception will be thrown when you are trying to locate a particular element on webpage that **is not currently visible even though it is present in the DOM**.
- Also, sometimes, if you are trying to locate an element with the XPath which associates with two or more element.



- There are two reasons for Stale element reference:
 - The element has been deleted entirely.
 - The element is no longer attached to the DOM.
- We face this stale element reference exception when the element we are interacting is destroyed and then recreated again. When this happens the reference of the element in the DOM becomes stale. So, we are not able to get the reference to the element.

SELENIUM



- [Components of Selenium](#)
- [Advantages of Selenium?](#)
- [Limitations of Selenium](#)
- [Challenges with Selenium?](#)
- [Which tests can be automated](#)
- [Which tests CANNOT be automated](#)
- [Locators](#)
- [XPath vs CSS](#)
- [Locate element using TEXT](#)
- [Absolute vs Relative Xpath](#)
- [Xpath syntaxes](#)
- [CSS syntaxes](#)
- [handle DYNAMIC ELEMENTS](#)
- [Handle in DYNAMIC TABLE](#)
- [Why I cannot find element?](#)
- [n'th child element using XPath?](#)
- [close\(\) and quit\(\)](#)
- [Implicit - Explicit - Fluent Wait](#)
- [Thread.sleep\(\)](#)
- [FindElement\(\) and FindElements\(\)](#)
- [Exceptions in Selenium](#)
- [Selenium Grid](#)

- [Hard Assert vs Verify \(Soft Assert\)](#)
- [driver.get\(\) & driver.navigate\(\).to\(\)](#)
- [Maximize & Resize](#)
- [Listeners on Selenium](#)
- [JavaScript Executor](#)
- [Dropdown in Selenium](#)
- [Screenshot in Selenium](#)
- [Tabs / Windows in Selenium](#)
- [Windows/OS popups](#)
- [Multiple windows in selenium?](#)
- [Alerts - Popups in Selenium](#)
- [Actions Class](#)
- [doubleClick & rightClick \(contextClick\)](#)
- [scroll down a page using JavaScript](#)
- [scroll down if the element is not visible](#)
- [Download in Selenium](#)
- [Uploading in Selenium](#)
- [Headless Browser](#)
- [Frames - iFrames](#)
- [Find all link present on the web page](#)
- [Https/SSL certificates Error Handle](#)
- [KeyBoard actions - Press Enter, sendK](#)

- [What is Selenium Framework](#)
- [Data Driven Framework](#)
- [Keyword Driven Framework](#)
- [HybridDriven Framework](#)
- [isDisplayed\(\), isEnabled\(\). And is Selected\(\) method](#)
- [DOM](#)
- [How to handle COOKIES?](#)
- [Position/location of the WebElement on the page?](#)

Components of Selenium



Selenium is a suite of tools for automated web testing. It is composed of;

- Selenium IDE(Integrated Development Environment); a Firefox plugin that works for recording and playing back.
- Selenium RC(Remote Control) (1.0) ; is a test tool and is used to work on JS to automate the web application.
- WebDriver (2.0); is a web automation framework and allows you to execute your tests in different browsers.
- Selenium Grid; allows tests to run in parallel across multiple machines.

Advantages of Selenium?



- Selenium is a suite of tools for automated web testing.
- Selenium is open source and free to use without any licensing cost
- It supports multiple languages like Java, Ruby, Python, C#...
- It supports multi browser testing
- It has a good amount of resources and helping community
- It supports many operating systems like Windows, Mac, Linux ...
- Interact with the web application

Limitations of Selenium



- Selenium supports testing of only web-based applications
- Mobile applications cannot be tested using Selenium
- Captcha and Barcode readers cannot be tested using Selenium
- Reports can only be generated using third-party tools like TestNG or JUnit.
- As Selenium is a free tool, thus there is no ready vendor support through the user can find numerous helping communities.
- The user is expected to possess prior programming language knowledge.

Challenges with Selenium?



- ***Sync issue***

- Sync issue or I would say timeout issue is one of the most challenging tasks in any test automation tool. If we do not handle sync issue then most of the script will fail. In one of the test survey, it as found that 80% of scripts fail due to improper sync while performing actions.
- We can avoid this by using smart wait which is present in Selenium like [implicit wait](#), [explicit wait](#), [fluent wait](#)

- ***Smart locators –locating elements***

- As we all know that locators are the core part of any scripting and We need to keep on enhancing our XPath and CSS for script stability because if [XPath](#) and [CSS](#) are not proper then it fails in upcoming releases.
- We should always write dynamic or custom XPath or class which can make our script more stable.

- ***Pop up handling***

- In many application, you will find random pop keeps coming and their behavior is not persistence so we also have to take care of these unwanted pop up which stops our execution.

Which tests can be automated



- functional tests (positive/negative, UI)
- smoke tests
- regression tests
- integration tests
- API
- Database
- end to end testing
- data driven

Which tests can not be automated



- Performance, load, stress testing, manual ad hoc testing, (These tests are done by experts trained in these tools)
- Pure database testing (if we only test the DB itself),
- Unit tests..., look and feel based testing (color, shapes, etc.),
- static testing
- Captcha is not automated as well.

Locators



- A locator is an address that uniquely identifies a web element within a web page. Selenium has several different types of locators to identify web elements in web pages. These include:
 - **ID** : *unique, but can be dynamic*
 - **Name** : *unique, but can be dynamic*
 - **Class name** : *Uses the class name attribute. if it has more than one word, separated by space, we can not use it*
 - **TagName** : *not unique. can be search in a search bar*
 - **LinkText** : *when locate links using their exact text. To locate any element using linkText, the tag of that element must be a and it must have a text*
 - **Partial Link Text** : *Uses partial link text to find web elements*
 - **CSS Selector** : *Works on element tags and attributes*
 - **Xpath** : *Searches elements in the DOM, Reliable but slow*

XPath VS CSS



Xpath	CSS
with Xpath, we can search elements backward or forward...	while Css works only in forward direction
works with text	does not support text
supports index: (//tag[@attribute='value'])[3]	does not support index
XPath has more combination so its powerful	Faster than xpath, easy to read and write,
<pre>//tagname[@attribute='value'][@attribute='value'] //tag[starts-with(@attribute, 'some value')] //tag[ends-with(@attribute, 'some value')] //tag[contains(@attribute, 'some value')] //tag[contains(text(), 'partial text')] or //tag[.='exact text'] //tagName[text() = 'linked_text'] //tagname[@attribute='value']/.. (parent) //tagname[@attribute='value']/parent::* (or tagname) //tagname[@attribute='value']/following-sibling::* //tagname[@attribute='value']/preceding-sibling::* //tag/tag/tag</pre>	<pre>tag[attribute = 'value'] tag[id='value'] --> tag#idValue or #idValue (# --> id) tag[class='value'] --> tag.classValue or .classValue (. --> class) tag[attribute^ = 'some value'] --> Starts-with tag[attribute\$ = 'some value'] --> Ends-with tag[attribute* = 'some value'] --> Contains tag[attribute = 'value'][attribute = 'value'] tag>tag>tag or tag tag tag</pre>

Locate element using text

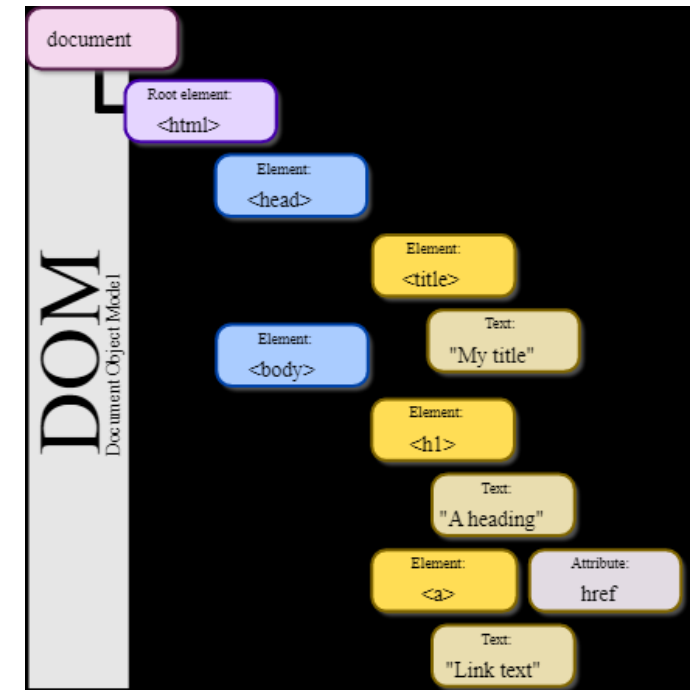


- The only locator that works with text is xpath.
- Matching exact text : **`//tag[.='text']`**
- Matching partial text : **`//tag[contains(text(), 'text')]`**

[back to main](#)

DOM

The Document Object Model (DOM) is a cross-platform and language-independent interface that treats an XML or HTML document as a tree structure wherein each node is an object representing a part of the document. The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree;



Absolute XPath vs Relative XPath



- **Absolute XPath** starts with single slashes (/) and it starts looking for element starting from the root element to child element.

Not stable **`/html/body/div/div/div/div/a`**

- **Relative XPath** starts with double slashes (//) and it is looking for element anywhere in the document.

It is more stable **`//body/nav//a`**

Xpath syntaxes



`//tag[@attribute='value']`

`(//tag[@attribute='value'])[5]` -> 5th WebElement

`//*[attribute='value']` -> any tag

`//tag[*='value']` -> any attribute

`//tag[@attribute='value'][@attribute='value']` -> and

To Locate Dynamic WebElement

`//tag[starts-with(@attribute, 'value')]`

`//tag[ends-with(@attribute, 'value')]`

`//tag[contains(@attribute, 'value')]`

`//tag[contains(text(), 'text')]`

`//tag[text() = 'usedId']`

`//tag[@attribute='value']/..`

-> parent element

`//tag[@attribute='value']/parent::*` or `parent::tagName`

-> parent

`//tag[@attribute='value']/following-sibling::*` (or tag)

-> following-sibling

`//tag[@attribute='value']/preceding-sibling::*` (or tag)

-> preceding-sibling

`//tag/tag/tag`

CSS syntaxes



`tag[attribute='value']`

`<button id="disappearing-button">Don't click,/button>`

`button[id=disappearing-button]`

`button#disappearing` -> **# stands for id** -> `#disappearing`

`a[class='nav-link']` -> `a.nav-link` -> `.nav-link`

To Locate Dynamic WebElement

`tag[attribute^='some value']` -> starts with

`tag[attribute$='some value']` -> ends with

`tag[attribute*='some value']` -> contains

`tag[attribute='value'][attribute='value']`

`tag>tag>tag`

handle dynamic elements



- Use explicit waits where necessary
- Find the static part of the element and write a locator (xpath or css)
- And then use `starts-with`, `contains`, `ends-with`, `text()`

To Locate Dynamic WebElement with XPath

<code>//tag[starts-with(@attribute, 'value')]</code>	-> starts with
<code>//tag[ends-with(@attribute, 'value')]</code>	-> ends with
<code>//tag[contains(@attribute, 'value')]</code>	-> contains
<code>//tag[contains(text(), 'text')]</code>	-> contains
<code>//tag[text() = 'usedId']</code>	-> text

To Locate Dynamic WebElement with CSS

<code>tag[attribute^='some value']</code>	-> starts with
<code>tag[attribute\$='some value']</code>	-> ends with
<code>tag[attribute*='some value']</code>	-> contains

Handle in dynamic table



- Use custom XPath and CSS locators
 - XPath: contains, starts with, ends with, contains text.
 - By finding the element in relation to another stable element using parent, child, preceding-sibling, following-sibling relationships
- I have utility methods that work with table. I have method that returns all the column names. I have a method that takes a table, number and returns all the data in that row.

How to handle Web Tables/grid?

- Table tag used for table data is arranged in a grid format

- th tag for column name Example –

```
<tr>
  <th>FirstName</th> column names on the very top row
  <th>Lastname</th>
  <th>Age</th>
</tr>
```

- </tr> tr tag used to indicate a row, applies to whole column td tag to indicate a column in a row Example

```
<tr>
  <td>Danny</td> actual_data_on_the_very_first_row
  <td>Smith</td>
  <td>29</td>
</tr>
```

- Some tables have tbody Used to indicate the data of the table, usually does not include column names (th)

Why I cannot find element?



- Locator changed
- There is an iframe
- Waiting time issue, page is loading slowly
- Element is dynamic, locator cannot find it. Use dynamic locator.
- Page is not fully loaded/opened. This is also waiting time issue
- Page changes and that element does not exist anymore

n'th child element using XPath?



There are two ways:

1. using square brackets with index position

`div[2]` will find the second div element

2. using position () method

`div[position()=2]` will find the second div element

close() and quit()



- `driver.close()` —> is used to close the current browser.
It only closes the current tab
- `driver.quit()` —> is used to close everything, all browser instances.
It closes all browsers, tabs, windows, every thing that was opened with selenium

Implicit Wait vs Explicit Wait vs Fluent Wait



- **Implicit wait** —> we use this wait to find elements in a specific time period that we defined. It works with findElement method. If locator cannot find the element in a determined time it throws “NoSuchElementException” exception. If find earlier it jump to the next step. We set it only one time using:

```
driver.manage().timeouts().implicitlywait(5, days)
```

- **Explicit wait** —> we use this wait for certain actions or conditions to happen. We call this every time when we want to wait. If locator can not find the element it throws “timeout” exception. If find earlier it doesn’t wait the whole time period and jump to the next step.

```
WebDriverWait wait = new WebDriverWait(driver, 5);  
wait.until(ExpectedConditions.visibilityOf(element));  
elementToBeClicable(WebElement) -- titleIs("text") -- titleContains("text");
```

- **Fluent wait** —> is used If the element is sometime visible and sometime invisible for a nonperiodic time frame.

```
Wait wait = new FluentWait(driver).  
    withTimeout(Duration.ofSeconds(30)).  
    pollingEvery(Duration.ofSeconds(2)).ignoring(Exception.class);
```

Thread.sleep()



- **Thread.sleep()** slows down selenium to catch up the webelement
- It throws exception, so we must handle the exception.

FindElement() and FindElements()



- **FindElement** returns first WebElement.

If the element not found, gives “NoSuchElement” exception.

- **FindElements** returns List of WebElement (List <WebElement>)

If the element is not found it returns NULL value (it does not throw Exception)

Exceptions in Selenium



- The most common exceptions in Selenium are:
 - **TimeoutException** is thrown when a command performing an operation does not complete in the stipulated time. To handle it I use EXPLICIT WAITS.
 - **NoSuchElementException** is thrown when an element with given attributes is not found on the web page. Check if locator is correct, Check if timing is correct, Check if element is hidden inside an iframe. To handle it I declare IMPLICIT WAIT
 - **ElementNotVisibleException**: This exception is thrown when the element is present in DOM (Document Object Model), but not visible on the web page
 - **StaleElementException**: This is thrown when the element is either deleted or no longer attached to the DOM. To handle it, I use try&catch block or refresh the page and use explicit wait.
- **NoSuchFrameException**
- **NoSuchAlertException**
- **NoSuchSessionException**

Selenium Grid



- We use Selenium Grid for Parallel Execution. It consist of one Hub VM + multiple Node VMs. Tests are actually executed on the Nodes, not on the Hub. Nodes can be Windows, Linux, Mac, also can have different browsers like: Chrome, Firefox, Edge, Safari etc.
- We use Docker Container via the EC2 machine on AWS to create Virtual Machines to run parallel execution. **don't mention Docker Container on the interview**
- I have only one HUB as a main centralized machine in order to run test cases. I create more than one NODES to execute test cases in different browsers. Each Node represents a different browser.
- Instead of WebDriver driver = new Chromedriver(); ==> I write;

WebDriver driver = new RemoteWebDriver(URL, capabilities)

- To start a HUB to run I write on the terminal that;

Java -jar selenium-standalone-<version of jar file (3.11.0.jar)>-role hub

If I want to add different port I write; -port 4441

We configure our BrowserFactory Class with options called RemoteDriver

- We pass IP address of the Selenium Grid Hub as desired capability
- ChromeOptions chrOptions = new ChromeOptions() ;
- driver = new RemoteWebDriver(new URL("http://34.235.0.4:4444/wd/hub"), chrOptions) ;}

Hard Assert vs Verify (Soft Assert)



HardAssert	SoftAssert (Verification)
<p>In HardAssert; when test fails; the test execution STOPS, and all test steps after that line of code are skipped.</p> <p>If there is, Selenium jumps the next Test method.</p>	<p>In SoftAssert; when verify test fails, the test does NOT stop, but it reports the failure regardless is true or false.</p> <p>It DOSEN'T STOP, but it reports the failure.</p>
<p>Assert class provides assertion methods</p> <pre>Assert.assertEquals(value1, value2) Assert.assertNotEquals(value1, value2) Assert.assertTrue(boolean condition) Assert.assertFalse(boolean condition)</pre>	<p>Soft assertions do not stop execution if test fails.</p> <pre>SoftAssert softAssert = new SoftAssert(); softAssert.assertEquals(value1, value2); softAssert.assertNotEquals(value1, value2); softAssert.assertTrue(value1, value2)...</pre>

driver.get() and driver.navigateto()



- **driver.get("url")** is used to open an URL and it will wait till the whole page gets loaded
- **driver.navigateto("url")** is used to navigate to an URL and it will **NOT** wait till the whole page get loaded

GUID

```
String mainGUID = driver.getWindowHandle();
See allGUIDs = driver.getWindowHandles();
driver.switchTo().window(guid);

driver.close();
○ driver.switchTo().window(mainGUID);
for(String guid : allGUIDs){

driver.switchTo().window(guid);
if(driver.getTitle().toLowerCase().contains("cnn")){
driver.findElement(By.id("dsk")).sendKeys("Michael");
break;
}
```

Maximize & Resize



- To maximize the size of browser window:

```
driver.manage().window().maximize();
```

- To resize browser Window :

- Create object of Dimensions class

```
Dimension newD = new Dimension(480,620);
```

- Resize the current window to the give n dimension

```
driver.manage().window().setSize(newD);
```

- If maximize() will NOT work, so I have a code for that :

```
ChromeOptions options = new ChromeOptions();  
options.addArguments("startmaximized");
```

Listeners on Selenium



- Listener is defined as interface that modifies the default TestNG's behavior. As the name suggests Listeners "listen" to the event defined in the selenium script and behave accordingly. Listeners allows customizing TestNG reports or logs.
- The types of Listeners in TestNG are,
 - IAnnotationTransformer
 - IAnnotationTransformer2
 - IConfigurable
 - IConfigurationListener
 - IExecutionListener
 - IHookable
 - IInvokedMethodListener
 - IInvokedMethodListener2
 - IMethodInterceptor
 - IReporter
 - ISuiteListener
 - ITestListener

JavaScript Executor



- JavaScriptExecutor is an interface that provides us to execute JavaScript code thorough Selenium driver.
- It provides “executeScript and executeAsyncScript” methods to run JavaScript in the context of the currently selected frame or window.

```
JavascriptExecutor js = (JavascriptExecutor) driver;  
js.executeScript(Script,Arguments);
```

Dropdown in Selenium



[also see this](#)

Select class is used to deal with drop down list in selenium. To create a select object we need to pass a webElement as constructor. That element must have the select tag

- select by Index: Takes a int param, selects based on the index 0 based.
- select by visible text: takes a string, select based on the text displayed.
- select by value: takes a string parameter selects based on the value attribute of the option

Example:

```
WebElement element = driver.findElement(by.id("dropdown"));
```

```
Select list = new Select (element)
```

```
list.getFirstSelectedOption;    -> select first the option
```

```
list.getOptions()               -> returns all the options
```

```
list.selectByVisibleText("text") -> select by text
```

```
list.selectByIndex(index)       -> select by index
```

```
list.selectByValue(value)       -> select by value
```

```
List<WebElement> options = list.getOptions(); //all options are in the list
```


Screenshot in Selenium



- I can take a screenshot by using the TakeScreenshot function.
- I can save that screenshot by using getScreenshotAs() method.
- Example:

```
File scrFile = ((TakeScreenshot)driver) .  
                getScreenshotAs(output Type.FILE);
```

Tabs / Windows in Selenium



- Selenium WebDriver handles one tab or window at a time. In order to control another tab we always need to switch to that tab.
- First of all, we need to determine the current window using the `driver.getWindowHandle()` method. This method returns a unique handle value.
- To be able to switch we need to get the all window handles using `driver.getWindowHandles()` method. I put all the handles to the Set Interface. And then in the for loop I compare the current window handle with each handle in the Set list. When I find the current window handle, I switch to that window.

- Handling Pop-up windows

- ```
String currentWindow = driver.getWindowHandle() -> returns a unique handle, save main window
Set <String> windows = driver.getWindowHandles(); -> returns all the tabs
 for (String handle : windows) {
 if (handle.equals(currentWindow)) {
 driver.switchTo().window(handle); } }
```

GUID

# Windows/OS popups?



- Selenium doesn't support windows-based apps, it is an automation testing tool that supports only web application testing. We could handle windows-based popups in Selenium using some third-party tools such as **AutoIT**, **Robot class**
- `driver.getWindowHandle()` —> This will handle the current window that uniquely identifies it within this driver instance.
- `driver.getWindowHandles()` —> To handle all opened windows

## Handling Pop-up windows

```
String currentWindow = driver.getWindowHandle() -> save main window
Set <String> windows = driver.getWindowHandles();
for (String handle : windows) {
 if (handle.equals(currentWindow)) {
 driver.switchTo().window(handle); } }
```

# Multiple windows in selenium?



- Selenium stays on one window. If you open a window and then 5 tabs popped open, selenium is focused on the first window
- If you are on a new window and you tell selenium to print an element on the default window, it will still work even that user's focus is on the new window. Must switch to new window:
  - Use `windowHandle()`
  - `Driver.getWindowHandle()`
    - Everytime Selenium opens a browser, it's going to give an index ID for the page called Handles
    - Returns the handle/id of current page (as a string)
  - `driver.switchTo().window(string handle)`
  - `driver.getWindowHandles()` for multiple windows

Returns a Set of window handles

- Switch using titles:

```
for(string handle: driver.getWindowHandles()){
 driver.switchTo().Window(handle)
 if(driver.getTitle().equals(targetTitle)
 break;
```

# Alerts - Popups in Selenium



If the alert on the browser comes from JavaScript, we use the Alert class.

We cannot locate alerts using inspect tools (id, name, css, xpath etc..)

Alerts are not html windows

## **Handling Alerts** using Alert Interface

```
Alert alert = driver.switchTo().alert();
```

```
alert.dismiss() -> click "Cancel" on popup
```

```
alert.accept() -> click "Ok" on popup
```

```
alert.getText() -> to get text from popup
```

```
alert.sendKeys("text") -> to send text
```

- When using Selenium, if alert shows up as part of the automation, we have to handle that alert before proceeding.
- Otherwise we get: **UnhandledAlertException**, and test exits.
- We can use Explicit wait / WebDriverWait to check whether the alert is there or not.
  - wait.until(ExpectedConditions.alertIsPresent ());

# Actions Class



Actions class is used for advanced mouse and keyboard interactions such as;

- Hover over element
- move to element
- Scroll up/down
- Double click
- Right click
- drag and drop
- keyboard combinations

## Actions Class

```
Actions actions = new Actions(driver);
```

actions

```
.moveToElement(elementOne) -> hover over an element
.pause(1000) -> Thread.sleep() between actions
.moveToElement(elementTwo) -> move to second element
.build() -> use when chaining actions
.perform() -> comes last in chain of actions
```

actions

```
.dragAndDrop(sourceElement, targetElement) -> drag & drop
.perform() -> comes last in chain of actions
```

actions

```
.clickAndHold(sourceElement)
.pause(1000) -> Thread.sleep() between actions
.moveToElement(targetElement) -> move to second element
.release() -> drop element
.build() -> use when chaining actions
.perform() -> comes last in chain of actions
```

actions

```
.sendKeys(Keys.ARROW_UP)
.sendKeys(Keys.ARROW_DOWN)
.sendKeys(Keys.PAGE_DOWN) -> to scroll down
.sendKeys(Keys.PAGE_UP) -> to scroll up
```

# Double click and Right click in Action class



- To perform any actions against web element using actions class, we need to locate the element first:

```
WebElement el = driver.findElement
```

- **Double Click (doubleClick):**

```
Actions actions = new Actions (driver).perform
actions.doubleClick(el).perform()
```

```
actions.moveTo(el).perform actions.doubleClick.perform
actions.moveTo(el).doubleClick().build.perform()
```

- **Right Click (contextClick):**

```
actions.contextClick(elementLocator).perform();
```

# scroll down a page using JavaScript in Selenium? 🏠

- We can scroll down a page by using `window.scrollTo()` function. Example:

```
((JavascriptExecutor)
driver).executeScript("window.scrollTo(0,500)");
```

- Or I use Action class:

```
Actions actions = new Actions(driver);
 actions.sendKeys(Keys.ARROW_UP)
 .sendKeys(Keys.ARROW_DOWN)
 .sendKeys(Keys.PAGE_DOWN) —> to scroll down
 .sendKeys(Keys.PAGE_UP) —> to scroll up
```



## scroll down if the element is not visible



```
WebDriver driver = new ChromeDriver();
JavascriptExecutor jse =(JavascriptExecutor)driver
.jse.executeScript('window.scrollTo(0,250)', '');
```

- OR, we can do as follows: →

```
jse.executeScript('scroll(0, 250);');
```

# Download in Selenium



- Selenium itself cannot verify file downloads, can click on download link but can't go outside the browser and open the downloaded file
- Other tools need to be used for that **Robot** and **AutoIT**

## Find dropdown list using getOptions

- We can find if a particular option is present in the dropdown list, using getOptions.

```
○ Select dropdown = new Select(dropdownElement);
List<WebElement> allElements = dropdown. getOptions();
boolean presenceOfElement = false;
for (WebElement element : allElements) {
```

```
String dropdownOptionValue = element. getText();
if (dropdownOptionValue. equals("Baku")) {
presenceOfElement = true;
}
}
```

- Alternatively, we can use xpath to find it as well:
  - driver.findElement (By.xpath("//select  
[@id='editor']/option[text()='Baku']" ));

# Uploading in Selenium



In order to upload file using selenium we need to locate the upload button in the DOM html. Then we do sendKeys by passing the path to the file.

- To upload file in selenium;
- First, I locate the element which takes the path of the file (Choose file button) :

```
WebElement input = driver.findElement("id");
```

- And then I provide the path to the file using the sendKeys method :

```
input.sendKeys("/path/to/file" + Keys.ENTER);
```

# Headless Browser



- Headless browser is a browser that does not open, it runs as a background service / program.
- I can do headless testing. One option is that in the runner class there is Dry keyword within the Cucumber Options. I make it “true” to run headless browser testing.
- The another way; I have a driver class that can open different browsers including headless browsers:

```
case "chromeHeadless":
```

```
 WebDriverManager.chromedriver().setup();
```

```
 driver = new ChromeDriver(new ChromeOptions().setHeadless(true));
```

```
 break;
```

```
case "firefoxHeadless":
```

```
 WebDriverManager.firefoxdriver().setup();
```

```
 driver = new FirefoxDriver(new FirefoxOptions().setHeadless(true));
```

```
 break;
```

# Frames - iFrames



- If we want to interact with elements inside frames/iframes, we have to switch to that frame first. Otherwise we get no such element exception. We cannot jump directly one iframe to another iframe. First we have to reach to parent or defaultContent frame.
- We can switch frames using 3 options.

`driver.switchTo().frame("id or name")` —> using the name or id of the element  
`.frame(index)` —> using the index of the frame starts with 0  
`.frame(webelement)` —> WebElement

## Handling iFrames

**driver**.switchTo().frame("id goes here") -> switches to frame  
**driver**.switchTo().parentFrame() -> switches back to parent frame  
**driver**.switchTo().defaultContent() -> switches back to original page

# Find all link present on the web page



- TagName should be “a”.
- I locate the webelements starts with tagname “a”.
- I also use driver.findElements instead of driver.findElement since the list of webelements will return.

```
List<WebElement> list = driver.findElements(By.tagName("a"));
```

# Https / SSL certificates Error Handling



- SSL (Secure Sockets Layer) is a standard security protocol for establishing secure connection between the server and the client
- Browser and the server use SSL Certificate mechanism to be able to establish a secure connection.
- SSL works through a combination of programs and encryption/decryption routine that exist on the web server computer and web server browser.
- When secure connection is not established between the server and client due to certificate SSL certificate error will occur
- Need to adjust our script in such a way that it will take care of SSL Exception/error by itself through Selenium Web driver.

- We need to create instance of DesiredCapabilities class :

- CHROME, IE → UI

```
DesiredCapabilities handlSSLErr = DesiredCapabilities.chrome ()
handlSSLErr.setCapability (CapabilityType.ACCEPT_SSL_CERTS, true)
WebDriver driver = new ChromeDriver (handlSSLErr);
```

- IE → UI

```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
System.setProperty("webdriver.ie.driver","IEDriverServer.exe");
WebDriver driver = new InternetExplorerDriver(capabilities);
```

- API                      Response response = given().relaxedHTTPSvalidation().when().get(baseUrl);

# sendKeys("text" + Keys.ENTER)



- To press Enter key using Selenium WebDriver,
- We need to use Selenium Enum keys with its constant Enter

```
WebElement button = driver.findElement(By.xpath("xpath"));
button.sendKeys("some text" + Keys.ENTER);
```

- How to input text in the text box without calling the sendKeys()?

```
//Use javascriptExecutor
JavascriptExecutor JS = (JavascriptExecutor)webdriver;
//To enter username
JS.executeScript("document.getElementById('User').value= 'www.google.com'");
//To enter password
JS.executeScript("document.getElementById('pass').value= 'tester'");
```

If you want to click **multiple keys** :

```
Actions act = new Actions(driver);
```

```
act.sendKeys(Keys.Ctrl + "a").build().perform();
```



# What is Selenium Framework



- Framework is the blueprint of test automation. It includes your folder structures, where to save you function library, test results, test data, resources. There are mainly 3 type of frameworks created by Selenium WebDriver to automate test cases:

## Data Driven Framework

- All of our test data is generated from some external files;
  - excel or
  - scenario outline in feature file or
  - TestNG Data Provider
- Selenium WebDriver is a great tool to automate web-based applications. But it does not support read and write operations on excel files. Therefore, we use third party APIs like Apache POI.

## Keyword Driven Framework

- Keyword driven testing is a scripting technique that uses data files to contain the keywords related to the application being tested.
- Keywords are written in some external files like excel file. Java code will call this file and execute test cases.

## HybridDriven Framework

- A combination of the DDF and KDF is commonly said to be HDF.
- Both the test data and test action are kept in external files.

# isDisplayed() - isEnabled() - isSelected() method



- **isDisplayed()** -> verifies the presence of a web element within the web page.  
If found -> true, If not found -> false  
checks for the presence of all kinds of web elements available
- **isEnabled()** -> verify if the web element is enabled or disabled within the web page.  
is primarily used with buttons
- **isSelected()** -> verifies if the web element is selected or not  
is used with radio buttons, dropdowns and checkboxes.

- To check Element Present:

```
if(driver.findElements(By.xpath("value")).size() != 0){
 System.out.println("Element is Present");
}else{
 System.out.println("Element is Absent");
}
```

- or

```
if(driver.findElement(By.xpath("value"))!= null){
 System.out.println("Element is Present");
}else{
 System.out.println("Element is Absent");
}
```

- To check Visible:

```
if(driver.findElement(By.cssSelector("a > font")).isDisplayed()){
 System.out.println("Element is Visible");
}else{
 System.out.println("Element is InVisible");
}
```

- To check Enable:

```
if(driver.findElement(By.cssSelector("a > font")).isEnabled()){
 System.out.println("Element is Enable");
}else{
 System.out.println("Element is Disabled");
}
```

- To check text present

```
if(driver.getPageSource().contains("Text to check")){
 System.out.println("Text is present");
}else{
 System.out.println("Text is absent");
}
```

boolean selectedOrNot = optionElement.isSelected();

# How to handle COOKIES?



- driver.manage().getCookies(); // Return The List of all Cookies
- driver.manage().getCookieNamed(arg0); //Return specific cookie according to name
- driver.manage().addCookie(arg0); //Create and add the cookie
- driver.manage().deleteCookie(arg0); // Delete specific cookie
- driver.manage().deleteCookieNamed(arg0); // Delete specific cookie according Name
- driver.manage().deleteAllCookies(); // Delete all cookies

## verify the position of the WebElement on the page?



- WebElement class has a getLocation method which returns the top left corner of the element

```
element.getLocation ();
```

# Page Factory class



- Page Factory class comes with Selenium.
- And it is used whenever we create page object classes.
- Its purpose is to initialize webElements that were defined in the class.

## Log4j

- I use Log4J for logging. I always log important steps in the test execution. That helps me to debug when there is a failure.
- Log4J is not a replacement for HTML reports.

Log4J has 3 components: 1) Loggers - responsible for logging information - we use only this as testers

2) Appenders - used to deliver LogEvents to their destination

3) Layouts - responsible for formatting logging information in different styles

To setup Log4J : add Apache Log4J dependency in pom.xml

create a new lod4j.properties file and save in the Project root folder

create an object - static final Logger logger = LogManager.getLogger(SelectHelper.class);

use that object to call logger methods: logger.info(); logger.warn(); logger.debug()



XML language, XML vs HTML

- [What is TestNG?](#)
- [What is assertions in TestNG?](#)
- [Difference between JUnit and TestNG](#)
- [Annotations in TestNG-1 \(@Test, @Method...\)](#)
- [Annotations in TestNG-2 \(others\)](#)
- [Ignore the Test](#)
- [How to EXCLUDE a particular test method from test execution?](#)
- [Cross Browser and Parallel Test in TestNG](#)
- [Group test using TestNG?](#)
- [Group of Groups in TestNG?](#)
- [Reports in TestNG](#)
- [@Factory and @DataProvider annotations?](#)
- [Rerun failed test cases in TestNG and JUnit](#)
- [How to run Cucumber with Testng?](#)
- [Excel in Framework](#)
- [Regular expression \(REGEX\) in testing.xml file to search @Test methods containing "smoke" keyword?](#)
- [@Test\(threadPoolSize=someInteger\)](#)
- [What does the Test TIMEOUT mean in testing?](#)



# What is TestNG

- TestNG is a centralized controller testing framework and allows us to manage run different test cases, then create reports, logs etc.
- Batch execution is also possible with TestNG. Let's say we have 100 test cases and TestNG can run them one by one.
- TestNG is also provide us Optional Execution opportunity. So we can skip some test cases (making them enable or disable; @Test(enabled=false))



# What is assertions in TestNG?

- **HARD ASSERT: Critical;** stop/failure Assert

It takes one boolean argument and String message. It Asserts that if the condition is true. If it isn't, test fails and stop running the code, it throws an AssertionError. If it there is another test method it jumps to this test method.

- **SOFT ASSERT: Non critical;** failure/continue SoftAssert

Soft Assert does not stop and does not throw an exception when an assert fails and It reports the fails and continue to run.



# Difference between JUnit and TestNG



| TestNG                                                                                                                                                                                                                                                                                                    | JUnit                                                                                |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| <b>TestNG Annotations:</b> @Test, @BeforeMethod, @AfterMethod, @BeforeClass, @AfterClass, @BeforeSuite, @AfterSuite, @BeforeGroups, @AfterGroups                                                                                                                                                          | <b>JUnit Annotations:</b> @Test, @Before, @After, @BeforeClass, @AfterClass, @Ignore |
| TestNG provide html report                                                                                                                                                                                                                                                                                | JUnit not                                                                            |
| TestNG has @DataProvider annotation which is same as Cucumber Scenario Outline for Data Driven Testing.                                                                                                                                                                                                   | JUnit not                                                                            |
| In TestNG, we can do parallel testing                                                                                                                                                                                                                                                                     | but JUnit doesn't support to parallel test, so we use sauceLab for it                |
| TestNG support group test                                                                                                                                                                                                                                                                                 | but JUnit doesn't support                                                            |
| TestNG and JUnit both of them have parameterize testing but TestNG parameterized test configuration is very easy to configure. <b>There are two ways to achieve parameterization in TestNG;</b> <ul style="list-style-type: none"><li>- @Parameters and TestNG xml file</li><li>- @DataProvider</li></ul> |                                                                                      |

# Annotations in TestNG-1



- **@Test** - actual test. Run in alphabetical order. By default priority=0. If we add priority, they will run in that order (lowest->highest).  
Hierarchy => **BeforeSuite > BeforeTest > BeforeGroups > BeforeClass > BeforeMethod > Test**
- **@AfterMethod** - methods with this annotation always run after the test method. Will execute after each method. Doesn't matter if test passes or fails {close browser, log out, delete test data, report close connections}
- **@BeforeMethod** - executes before each method {prepare test data, set path, open browser, create connections, initialize classes, open url, login}
- **@BeforeClass** - executes once in the beginning {prepare test data, set path, open browser, create connections, initialize classes, open url}
- **@AfterClass** - executes once in the end {close browser, log out, delete test data, report close connections}
- **@Ignore** - ignores Test Case

A large blue arrow pointing to the right, containing the text "Next slide".

Next slide

# Annotations in TestNG-2



- (enable = true/false) → *true: executes the test case, false: ignore the test case*
- (timeout = int milliseconds) → *gives time limit to the test case*
- (priority = int level) → *deciding order of execution*
- (description = "text") → *is used for the test description*
- (expectedexception = Exception) → *is used for unchecked exceptions*
- DependsOnMethods = "test method name" *You can add multiple test names. If the first one fails, the 2nd test won't run at all*
- (invocationCount = int count) → *decides how many times the test cases should be executed :*

```
@Test(invocationCount=10) //runs 10 times
 Public void testcase() {
 // to do
 }
```



# Ignore the Test in TestNG

To ignore the test case, we use the (enabled = false) parameter :

```
@Test(enabled=false)
```

Example:

```
@Test(enabled=false)
public static void TestX() {
 // to do
}
```

# How to EXCLUDE a particular test method from test execution?



- By adding the exclude tag in the testing.xml

```
<classes>
 <class name="TestCaseName">
 <methods>
 <exclude name="TestMethodNameToExclude"/>
 </methods>
 </class>
</classes>
```



# Cross Browser and Parallel Test in TestNG

- In my previous project I used testng.xml file for cross browser testing.
- Basically, inside the suite there are 3 keys (name, thread count, parallel) and I created 2 different tests, one of them is for Chrome and the other one is for Firefox.
- There is also parameter annotation and include name and value; name is browser and value is Chrome.

- In xml file write;  
parallel="tests" thread-count="4"
- Thread-count is how many browser do you want to open at the same time
- In xml file you can add .\* to run everything
- Ex: <package name=".\*"></package>

```
<?xml version="1.0" encoding="UTF 8"?>
<!DOCTYPE suite SYSTEM ...>
<suite ...>
 <test name="ChromeTest" ... >
 <parameter name="browser" value="chrome"/>
 <classes>
 <class name="testsuite..."/>
 </classes>
 </test> <!-- First Test -->
 <test name="FireFox" ... >
 <parameter name="browser" value="FireFox"/>
 <classes>
 <class name="testsuite..."/>
 </classes>
 </test> <!-- Second Test -->
</suite> <!-- Suite -->
```



# Group test using TestNG?

```
@Test (groups={"smokeTest", "functionalTest"})
public void loginTest() {
 System.out.println("Logged in successfully");
}
```

[back to main](#)

[XML language, XML vs HTML](#)

[XML](#) stands for eXtensible Markup Language. It's a markup language like HTML.

It was designed to store and transport data (like JSON). XML does not do anything, its just information wrapped in tags

[HTML](#) - is about [displaying content](#) (information)

[XML](#) - is about [storing and transporting](#) content (information)

XML tags are defined by author (not predefined like HTML tags are)



# Group of Groups in TestNG?

- These groups are called metagroups.
- Example:

You might want to define a group all that includes smokeTest and FunctionalTest. Let's modify our **testng.xml** file:

```
<groups>
 <define name="all">
 <include name ="smoke Test"/>
 <include name = "functionalTest"/>
 </define>
 <run>
 <include name = "all"/>
 </run>
</groups>
```





# Reports in TestNG?

- TestNG offers two ways to produce a report
- **Listeners** are notified in real time of when a test starts, passes, fails, etc...
- **Reporters** are notified when all the suites have been run by TestNG.
- The IReporter instance receives a list of objects that describe the entire test run.
- In TestNG, Extent is also an HTML reporting tool which gives detailed test steps and screenshots. It also provides metrics on the test results.

**ExtentReports** class is used to start and build the reports. Only 1 instance is needed.

**ExtentHtmlReporter** class creates HTML report file.

**ExtentTest** defines a test, enables adding logs, authors to the test etc.

**XML runner** is used to run multiple tests from different classes

1. Right click on project name
2. New File -> testng\_runner.xml

# @Factory and @DataProvider annotations?



- **@Factory** executes all the test methods present inside a test class using a separate instance of the class with different set of data.
- **@Factory** is declared in a different class from the **@BeforeClass** and **@TestMethod**. So **@BeforeClass** runs everytime whenever a data object called from the class where **@Factory** in it.
- **@DataProvider** is a test method that uses DataProvider will be executed the specific methods multiple number of times based on the data provided by the DataProvider. **@DataProvider** is declared in the same class with **@BeforeClass**, **@TestMethod**. So **@BeforeClass** runs only one time.
- By using **@DataProvider** annotation, we can create a Data Driven Framework.

# @Factory and @DataProvider annotations?



```
public class SimpleTest
{
 private String param = "";

 public SimpleTest(String param) {
 this.param = param;
 }

 @BeforeClass
 public void beforeClass() {
 System.out.println("Before SimpleTest class executed.");
 }

 @Test
 public void testMethod() {
 System.out.println("testMethod parameter value is: " + param);
 }
}

public class SimpleTestFactory
{
 @Factory
 public Object[] factoryMethod() {
 return new Object[] {
 new SimpleTest("one"),
 new SimpleTest("two")
 };
 }
}
```

Let's run the above test.

```
Before SimpleTest class executed.
testMethod parameter value is: two
Before SimpleTest class executed.
testMethod parameter value is: one
PASSED: testMethod
PASSED: testMethod
```

As you can see from the previous test results, the beforeClass() method is executed before each execution of testMethod(). This shows that factory implementation executes the test method for each individual instance of the test class.

```
public class DataProviderClass
{
 @BeforeClass
 public void beforeClass() {
 System.out.println("Before class executed");
 }

 @Test(dataProvider = "dataMethod")
 public void testMethod(String param) {
 System.out.println("The parameter value is: " + param);
 }

 @DataProvider
 public Object[][] dataMethod() {
 return new Object[][] { { "one" }, { "two" } };
 }
}
```

Let's run the above test.

```
Before class executed
The parameter value is: one
The parameter value is: two
PASSED: testMethod("one")
PASSED: testMethod("two")
```

As you can see from the preceding test results the class beforeClass() is executed only one time irrespective of how many times the test method is executed.



# Rerun failed test cases in *TestNG* and *JUnit*

## ***By using TestNG***

1. After the first run of an automated test run. Right click on Project - click on Refresh
  2. A folder will be generated named “test-output” folder. Inside “test-output” folder, you could find “testng-failed.xml”
  3. Run “testng-failed.xml” to execute the failed test cases again.
- If you have three test cases and all the test cases are executed successfully, that means you are not able to see this folder under the test-output folder. This folder will appear only when if there is a test case that failed. Then run this file, it will run only failed test cases.

By Using JUnit



# Rerun failed test cases in *TestNG* and *JUnit*

## ***By using JUnit***

- First, I add this line to the cukesrunner, in the `plugins` option: "rerun:target/rerun.txt"  
**rerun** creates a text file with list of failed scenarios  
**target/rerun.txt** --> location and file name
- Second, I create a new runner class
  1. Right click on **runners** package
  2. New --> java class
  3. Name: `FailedTestRunners.java`
  4. OK
- Then I add the features options to the new runner file as like:  
**features**="@target/rerun.txt" This means, run all the scenarios listed in the rerun.txt file
- I also change the report file path to avoid overriding the success reports.
- Then I add the Cukes runner and the failed test runner to the pom file inside `<include>` tag.



# How to run Cucumber with TestNG?

- Add cucumber-testng maven dependency to pom.xml file (version 6.14.3)

```
<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
 <groupId>org.testng</groupId>
 <artifactId>testng</artifactId>
 <version>6.14.3</version>
 <scope>test</scope>
</dependency>
```

- Make CukesRunner extend to AbstractTestNGCucumberTests
  1. cucumber-testing - > pom.xml
  2. CukesRunner extends AbstractTestNGCucumberTests{ }



# Excel in Framework

- Apache POI Used to read data from external sources such as excel, csv, text files.
- 1st dependency is used to connect to Microsoft files.
- 2nd - to connect to new office.
- I past the excel file in the 'resources' directory. Then I create a class and 3 objects:

```
//takes the file path and creates connection to the file
private FileInputStream fileInputStream;
//represents excel file
private Workbook workbook;
// represents a single sheet
private Sheet workSheet;
```

Create @BeforeTest

```
@BeforeTest
public void setUp() throws IOException {
 // location of the excel file
 String filePath = "src/test/resources/Countries.xlsx";
 // create input stream using file path
 fileInputStream = new FileInputStream(filePath);
 // create the workbook object
 workbook = WorkbookFactory.create(fileInputStream);
 //create a worksheet by it's index
 workSheet = workbook.getSheetAt(0); }
```

7. Create @Test

```
// getSheetName --> returns the name of the current sheet
String sheetName = workSheet.getSheetName();
// getLastRowNum --> returns the last row num
int rowCount = workSheet.getLastRowNum();
// getRow(0) --> get the first row
// getLastCellNum --> index of last cell(number of columns)
int colCount = workSheet.getRow(0).getLastCellNum();
```

In order to update excel file we need to change cell value and update the excel file.

```
Cell colName = sheet.getRow(i).getCell(j) -> target cell
colName = sheet.getRow(i).createCell(j) -> creates a cell
colName.setCellValue("text") -> changes value of the cell
```

```
FileOutputStream fileOutputStream = new FileOutputStream(filePath);
workbook.write(fileOutputStream); -> to update actual excel file
```

```
fileInputStream.close();
fileOutputStream.close(); -> always close streams
```

## How to write regular expression in testing.xml file to search @Test methods containing "smoke" keyword?



- Regular expression to find @Test method containing keyword "smoke" is a mentioned below

```
<methods>
 <include name=".*smoke.*"/>
</methods>
```



## What is the use of @Test(threadPoolSize=someInteger)?



- The threadPoolSize attribute tells to from a thread pool to run the test method through multiple threads
- Note: this attribute is ignored if invocation count IS NOT SPECIFIED

# What does the test timeout mean in testing?



- The maximum number of **milliseconds** a test case should take

```
@Test1(threadPoolSize=3,invocationCount=10,timeOut=10000)
public void test() {}
```

- In this example:
  - The function **Test1** will be **invoked 10 times** from **3 different threads**,
  - Additionally, a **timeout of 10 seconds (10000milliseconds)** guarantees that none of the threads will block on this thread forever.



- [What is Cucumber](#)
- [Components of Cucumber BDD framework](#)
- [What is Gherkin](#)
- [Runner Class](#)
- [Report in Cucumber](#)
- [How to see your reports in cucumber?](#)
- [dryRun in Cucumber](#)
- [Hooks in Cucumber](#)
- [Re-run Failed Tests in Cucumber](#)
- [Re-run Failed Tests in JENKINS](#)
- [How do you take screenshots in cucumber](#)
- [Maven Lifecycle](#)
- [POM.xml File](#)
- [Scenario Outline vs Scenario?](#)
- [Page Object Model \(POM\)](#)
- [Page Factory?](#)
- [@RunWith & @CucumberOptions](#)

- [How to run Cucumber with JUnit?](#)
- [How to run Cucumber with TestNG?](#)
- [How to run a Cucumber with DDT?](#)
- [What is Background?](#)
- [How do I limit the types of variables I can pass?](#)
- [DDT - Data Driven Testing in Cucumber](#)
- [DDT - Data Driven Testing in TestNG](#)
- [How do I use cucumber scenario for DDT?](#)
- [How to use Map in cucumber?](#)
- [How to use POJO in cucumber?](#)
- [EXCLUDE a particular test method from test execution](#)
- [Singleton in Framework](#)
- [How does feature file work?](#)
- [Parallel Test in Cucumber](#)

# What is Cucumber



- Cucumber is a testing tool which is used in Behavior Driven Development.
- Cucumber works with JUNIT and TESTNG. Cucumber is good for non technical people, easy reading, saves time, reusability, maintainable, default report, easy to create smoke tests, regression using tags.
- One of its wonderful main features is the ability to execute plain text functional description (written in language named Gherkin) as automated tests.
- Writing BDD tests in Ubiquitous language, a language structured around the domain model and used by all team members including developers, testers, BAs, etc.
- Cucumber BDD testing tool builds bridges between the technical and nontechnical members of a software team.
- Last but not least, Cucumber is an Automated Acceptance Test Tool which running tests written in a Behavior Driven Development (BDD) style.

# Components of Cucumber BDD framework?



- **Feature files**

Consists of scenarios that test a certain feature or functionality

Feature is main story while scenarios are the test cases to the story(feature)

- **Pages Package**

All the pages in my application are represented in this package. I locate the WebElement in these pages.

- **Cukes Runner**

A class that strictly runs the tests, generates codes for step definition @smoketest

Cukesrunner → IN CUCKESRUNNER I HAVE A FEATURE LOCATION THAT SHOWS WHERE MY FEATURE ARE LOCATED

- **Step definition**

A class that made of steps that starts with Gherkin language. Make sure the step definition is in the same package as cukesRunner, or child package (not parent or sibling)

# What is Gherkin



- GHERKIN is a language used by feature files
- Feature
- Scenario
- BackGround
- Scenario Outline - Examples
- Given
- When
- Then
- And
- But

# Runner Class



- I have a runner class that runs my feature files. Using runner class I can generate step defs, run certain tests, configure reports etc.

```
@RunWith(Cucumber.class) //RunWith comes from JUnit and triggers the execution of the test
@cucumberOptions(
 plugin = {"html:target/default-cucumber-reports", //to generate report in html format
 "json:target/cucumber.json", //to generate report in JSON format
 "rerun:target/rerun.txt" //this is for the failed test report
 },
 features = "src/test/resources/com/vytrack/features/", //path to feature file
 glue = "com/vytrack/step_definitions", //path to step definitions classes
 tags = "@wiper",
 dryRun = false //to check the mapping is proper between feature file and step def file. (true, false)
)
public class CukesRunner {

}
```

# Report in Cucumber



- In my framework I can generate html and JSON reports. My reports have detailed steps and the screenshot for failures in *the HOOK Class*.
- **Default cucumber html reports.** —> *this is a default reporter meaning I do not need to do anything in the pom file to get this.*

We get this report every time when we run the cukesRunner. It does not depend on terminal or maven. Even if I run it by right clicking the cukesRunner or clicking on that green thingy, we still get the report.

- **maven-cucumber-reporting** —> *this is a plugin in pom file. I have to add this plugin info to my pom file to make it work.* We also need to add JSON option into the cukesRunner under the plugin. This report shows more metrics, pass, fail rates, enables sorting by tags.
- To generate this plugin we always have to run using terminal or maven

Next: Maven Cucumber Report



# Report in Cucumber



**Maven cucumber reporting** → third party open source reporting tool for cucumber. Provides better interface and more metrics about the test results.

1. Add new plugin option to CukesRunner

```
plugin = { "html:target/default-cucumber-reports",
 "json:target/cucumber.json"},
```

2. Add the plugin to pom file

<https://github.com/damianszczepanik/maven-cucumber-reporting>

3. Make sure name of the json file matches one provided in param

```
<param>**/cucumber*.json</param>
```

4. Make sure name of the runner matches one provided in include

```
<include>**/CukesRunner.java</include>
```

5. Run mvn verify in IntelliJ terminal every time we need a report (not generated if we run from IntelliJ)

Cucumber 4 supports parallelization out of the box. It does it using maven plugin: maven-surefire-plugin

```
<include>**/runners/*ParallelRunner*.java</include>
```

Include tag provide names of classes we want to run.

# How to see your reports in cucumber?



- Our Cucumber BDD framework generates default HTML reports.
- The report shows the pass/fail coverage for feature files, tags, steps
- The report contains all the steps for each test. The report has screenshots for failures
- My framework generates cucumber reports in the target folder which contains the reports. When we run the tests on Jenkins, Jenkins saves the report of every run. Home page of the Jenkins job always points to the last run reports. All the reports for previous runs can be found under the build number.

Go to target folder

Open with system explorer

Go to target>cucumberreport>index shows the tests you ran

# dryRun in Cucumber



- dryRun is used when we want to generate step definitions without actually running tests. If it is true, browser will not open, only the codes will be executed.

# Hooks in Cucumber



## What are Hooks in cucumber?

- Cucumber hook allows us to better manage the code workflow and helps us to reduce the code redundancy. We can say that it is an unseen step, which allows us to perform our scenarios or tests.
- Class that uses
  - @Before → runs before each cucumber scenario
  - @After → runs after each scenario (It will always run no matter if scenario passes or fails)
- Hook Class must be in same package as stepDefinition
- I implemented [SCREENSHOTS inside HOOK class](#)
- Hook Class will not run if dryRun=true
- I use Scenario as a parameter in my before/after method

# How do you take screenshots in cucumber?



## How do you take screenshots in cucumber?

- In my @Aftermethod I have a code:
- I use TakeScreenShot interface
- You can store screenshot as a byte or file

*How to generate screenshots → add Scenario object as a parameter to the After hook method. Using this object we check if scenario passed/failed. If failed we adding the screenshot to html report.*

```
@After ()
public void tearDown(Scenario scenario) {
 // check if scenario failed
 if (scenario.isFailed()) {
 // take the screenshot
 final byte[] screenshot = ((TakesScreenshot) Driver.get()).getScreenshotAs(OutputType.BYTES);
 // attach the screenshot to the report
 scenario.embed(screenshot, "image.png"); }
 Driver.closeDriver();}
```



# Re-run Failed Tests in Cucumber



- We have to report all the failed tests using rerun option. When we run tests, all the failed scenarios will be reported in the **rerun.txt file**.
- I create another **FailCukesRunner class** will only runs tests listed in the **rerun.txt file**.
- I add both of the runners in the pom.xml file and run tests using maven from terminal. Then my pom file runs the main CukesRunner first, after that it runs the failed test runner second.
- We use the re-run option in the CukesRunner.
  - I add the rerun to cukesRunner.
  - This option will create a file with a list of failed tests
- I create a second runner class which points to file with a list of failed tests
- I add the second runner in the pom file

# Re-run Failed Tests in JENKINS



- In Jenkins there are plugin that re run the failed tests Unit cases.
- So you can configure your Maven build execution on Jenkins using the option:

`Dsurefire.rerunFailingTestsCount=2`





## Different Phases in Maven Build Lifecycle

- validate** Validate the project is correct and all necessary information is available.
- compile** Compile the source code of the project.
- test** Test the compiled source code using a suitable unit testing framework.
- package** Take the compiled code and package it in its distributable format.
- verify** Run any checks on results of integration tests to ensure quality criteria are met.
- install** Install the package into the local repository, for use as a dependency in other projects locally.
- deploy** Copy the final package to the remote repository for sharing with other developers and projects.



# POM.xml File



- A file that manages the whole project
- When you run a maven command, everything should be done through the pom.xml

# Scenario Outline vs Scenario?



- Scenario in cucumber runs once.
- Scenario Outline is used for data driven testing  
Have the same cucumber steps but we provide data after the scenario as a table using keyword EXAMPLES

# Page Object Model (POM)



4

→ What is POM (Page Object Model)?

→ What are its advantages?

- Page Object Model is a design pattern to create Object Repository for web UI elements.
- Each web page in the application should have corresponding page class.
- Page class will find the WebElements of that web page and also contains Page methods which perform operations on those WebElements.

## **Advantages:-**

- Keep operations and flows in UI separate from Verification – clean & easy to understand code
- Object Repository independent of Test Cases – multiple tests use same Object Repository
- Reusability of code

# Page Factory?



- Page Factory is an inbuilt Page Object Model concept for Selenium WebDriver which is very optimized.
- It allows separation of Page Object Repository and Test Methods.
- Page Factory class is a class that is used to initialize the page object classes.
- It provides `@FindBy` annotation to find the WebElements.
- Without using the `PageFactory.initElements`, page object class will not work as expected, for example `@FindBy` will not work.

constructor -->

```
public NavigationBar(){
 PageFactory.initElements(Driver.get(), page: this);
}
```

# @RunWith & @CucumberOptions



**@RunWith**(Cucumber.class) *//RunWith comes from JUnit and triggers the execution of the test*

**@CucumberOptions**(

plugin = {"html:target/default-cucumber-reports", *//to generate report in html format*

"json:target/cucumber.json", *//to generate report in JSON format*

"rerun:target/rerun.txt" *//this is for the failed test report*

},

features = "src/test/resources/com/vytrack/features/", *//path to feature file*

glue = "com/vytrack/step\_definitions", *//path to step definitions classes*

tags = "@wiper",

dryRun = **false** *//it can be true or false. When dryRun=true, Hook Class and any browser will not run.*

)

**public class** CukesRunner {

}

- Just like TestNG was providing Groups to run specific test cases together, Cucumber gives that option using Tags.
- We can tag scenarios like @Smoke, @Regression, @CustomName
  - @Regression @Smoke
  - Scenario: Add item to cart
  - Given I am on homaepage
- We can add multiple tags to one scenario
- We can run scenarios that have any of the mentioned tags (OR logic):
  - tags = {"@search", "@smoke" }
- We can also run scenarios that have all the mentioned tags. (AND logic):
  - tags = {"@search", "@smoke" }
  - Difference: each tag is in its own "" quotes.
- We can also ignore running certain tags
  - tags = {"@regression", "~@wip" }

# How to run Cucumber with JUnit?



## How to run Cucumber with JUnit?

- Add cucumber JUnit dependency
- Adding `@RunWith (Cucumber.class)` on top of `cukesRunner` class

# How to run Cucumber with TestNG



## How to run Cucumber with TestNG?

- Add cucumber **TestNG** dependency
- Make CukesRunner extend to AbstractTestNG CucumberTests

# How to run a Cucumber with DDT?



- I use Cucumber tables and also Scenario Outline with Examples:

[| Home](#) | [Emails](#) | [Documents](#) | [Projects](#) |

- You get the method with (DataTable arg1)
- In the parameter DataTable you can change it to

`List<YourType>`, `List<List<E>>`, `List<Map<K,V>>`, and `Map<K,V>`

- Prints in order for list
- No order for map

```
Feature: Login as different people using maps
@wiper
Scenario Outline: Verify title
 Given I login using these credentials
 | username | <username> |
 | password | <password> |
 When I navigate to "Customers" "Contacts"
 Then the page title should be "All - Contacts - Customers"
Examples:
 | username | password |
 | salesmanager101 | UserUser123 |
 | salesmanager102 | UserUser123 |
 | salesmanager103 | UserUser123 |
```



# What is Background?



- Cucumber has their own before method
- The one in hooks is for java
- A step that runs BEFORE a scenario inside the feature file
- Can only put on top, before all scenarios
- Cannot put pipelines in backgrounds (Only in scenario outline)

**Feature:** User account information

**Background:**

**Given** I login as a "driver"

**And** I navigate to "Customers" "Contacts"

**Scenario:** test with manager

**When** I click on customer with email "odugmore5@sakura.ne.jp"

**Then** customer email should be "odugmore5@sakura.ne.jp" in the account page

**Scenario:** test with admin user

**When** I click on customer with email "mbrackstone9@example.com"

**Then** customer email should be "mbrackstone9@example.com" in the account page

# How do I limit the types of variables I can pass?



- In the gherkin parenthesis you can add (Collaboration | Sales | Marketing, etc.)
- Ex:

@When("^I hover over the (Collaboration | Sales | Marketing | Activities | All ) menu\$")

```
public void i_hover_over_the_Collaboration_menu(String menu) {
 switch(menu) {
 case "Sales":
 BrowserUtils.hover(dashboard.sales); break;
 case "Marketing":
 BrowserUtils.hover(dashboard.marketing); break;
 case "Collaboration":
 BrowserUtils.hover(dashboard.collaboration); break;
 case "Activities":
 BrowserUtils.hover(dashboard.activities); break;
 case "All":
 BrowserUtils.hover(dashboard.all); break;};
}
```

## What if you have a scenario that has two parameters (limiting parameter, table parameter)?



- Example :
  - Scenario: Verify Collaboration menu options
  - Given I logged into suiteCRM
  - When I hover over the Collaboration menu
  - Then the following menu options should be visible for Collaboration:  
| Home | Emails | Documents | Projects |
  - In this scenario i have a table, I want to limit collaboration to just collaboration and the other menus categories
- Solution:
  - @Then("^following menu options should be visible for  
( Collaboration | Sales | Marketing | Activities | All ):\$")
  - public void following\_menu\_options\_should\_be\_visible\_for\_Collaboration(String menu, List<String> options) {
  - String menu represents the 5 menu options ( Collaboration | Sales | Marketing | Activities | All )  
List<String>options represents the tables; | Home | Emails | Documents | Projects |

# DDT - Data Driven Testing in Cucumber



- Test data is separated from code and stored into external sources:
  - Cucumber Examples table
  - Excel files, CSV files
  - Database
- If the amount of data is not that huge, then I use Cucumber Scenario outline with Examples table.
- And other times I maintain test data in Excel files, and I use Apache POI library to read and write data
- If data comes from a database, or I need to do database validation, I use SQL queries along with JDBC library in java.

## Data Driven Testing

- **WHEN:** Whenever a functionality or a module in an app requires testing with multiple sets of data(Parametrization), Multiple inputs then we need to perform data driven testing and automation.
- These scenarios are one of the things That must be automated.
- **HOW:** Test data is separated from code and stored into external sources: Cucumber Examples table, Excel files, CSV files, Database.
- **BENEFIT:** More organized, Data centralized, Collaboration on test data - it can come from BA, MTs etc|

# DDT - Data Driven Testing in TestNG



- By using **@DataProvider** annotation, we can create a Data Driven Framework

```
@DataProvider(name="getData") Public Object[][] getData(){ Object [][] data = new Object[2][2];
Data[0][0] = "firstUid"; Data[0][1] = "FirstPWD";
Data[1][0] = "SecondUid";
Data[1][1] = "SecondPWD"; Return data; }
```

## *Extra Information:*

### **Data Driven Testing**

- **WHEN:** Whenever a functionality or a module in an app requires testing with multiple sets of data(Parametrization), Multiple inputs then we need to perform data driven testing and automation.
- These scenarios are one of the things That must be automated.
- **HOW:** Test data is separated from code and stored into external sources: Cucumber Examples table, Excel files, CSV files, Database.
- **BENEFIT:** More organized, Data centralized, Collaboration on test data - it can come from BA, MTs etc|

# How do I use cucumber scenario for DDT?



- In my current project I use Scenario Outline with Examples
- In my scenario feature file, whenever I'm using a `<variable>` as a data driven, I use "`<variable>`"
- Then in Examples:

## Examples:

variable	-> column name
data1	-> row1
data2	-> row 2
data3	-> row3

**Feature:** Login different types of users

**Scenario Outline:** Login as different users

**Given** I login as a "`<user>`"

**When** I logout

**Then** the page title should be "`Login`"

**Examples:**

user
driver
sales manager
store manager



# How to use Map in cucumber?



- Using a nonScenario Outline
- Scenario: Create contact using a map
  - Given I logged into suiteCRM
  - When I create a new contact:

first_name	John
last_name	Smith
cell_phone	801 888 8889
  - Then I should see contact information for "John Smith"
  - Left side is key, and right is value 2 columns only
- Using a Scenario Outline
  - Scenario Outline: Create contact using a map
  - Given I logged into suiteCRM
  - When I create a new contact:

first_name	<first_name>
last_name	<lname>
cell_phone	<cell_phone>
office_phone	<office_phone>
  - Then I should see contact information for "<first\_name> <lname>"
  - Examples:

first_name	lname	cell_phone	office_phone
Michael	Jackson	1234567890	2345678891
Bonnie	Garcia	4569871234	4567890987
- In step def I write;

```
@When("^I create a new contact:$")
public void i_create_a_new_contact(Map<String,String>contact) {
 // open the create contact dialog
```

# How to use POJO in cucumber?



- I Create contactBean class
  - I Add all private variables
  - I Add the getter/setters methods
- I Create bean feature file
- I Create a table with first row containing the variables in the contactBean class
  - I Add values under the table
  - I Implement method with parameter (List<ContactBean>contacts)
- I write the *Scenario*: Create contact
  - *Given* I logged into suiteCRM
  - *When* I save a new contact:

I firstName	I lastName	I officePhone	I cellphone	I email
I Metin	I Kaya	I 3456758888	I 1234329999	I metinKaya@gmail.com
  - *Then* I should see contact information for "Steve Gates"



# How to EXCLUDE a particular test method from test execution?



- By adding the exclude tag in the testing.xml

```
<classes>
 <class name="TestCaseName">
 <methods>
 <exclude name="TestMethodNameToExclude"/>
 </methods>
 </class>
</classes>
```

# Singleton in Framework



```
import ...
```

```
//Singleton class is used to create unique driver.
```

```
//So we cannot use singleton pathern when we run parallel testing
```

```
//To create a singleton class pathern, we need to do following steps:
```

```
// 1. Declare constructor of class as private so that no one instantiate class outside of it.
```

```
// 2. Declare a private static reference variable of class. Static is needed to make it available globally.
```

```
// 3. Declare a static method with return type as object of class which should check if class is already instantiated once.
```



```
public class Driver {
```

```
 private Driver() {} //1. Declare constructor of class as private so that no one instantiate class outside of it.
```

```
 private static WebDriver driver; // 2. Declare a private static reference variable of class.
```

```
 public static WebDriver get() { //3. Declare a static method with return type as object of class
```

```
 if (driver == null) {
```

```
 String browser = ConfigurationReader.get("browser");
```

```
 System.out.println("browser = " + browser);
```

```
 switch (browser) {
```

```
 case "chrome":
```

```
 WebDriverManager.chromedriver().setup();
```

```
 driver = new ChromeDriver();
```

```
 break;
```

```
 case "chromeHeadless":
```

```
 WebDriverManager.chromedriver().setup();
```

```
 driver = new ChromeDriver(new ChromeOptions().setHeadless(true));
```

```
 break;
```

# How does feature file work?



- Feature → description of what is being tested.
  - @tags, Sample feature file
  - Background runs before both of the scenarios
- Scenario → description of the scenario being test
  - Given I am on the login page
  - And I enter username and password
  - When I click on the submit button
  - Then I should be able to see the Dashboard page
- Given → a precondition
- When → condition that triggers the expected result
- Then → expected condition

# Parallel Test in Cucumber



- I use **maven-surefire-plugin**. This plugin executes tests in parallel. In this plugin configuration, we indicate which runner files we want to run. We can also indicate how many simultaneous tests we want to run.

```
<include>**/runners/*TestRunner*.java</include>. → plugin will run these files
<threadCount>10</threadCount> → this shows how many browsers we want to have in at the same time.
<parallel>classes</parallel> → this line tells that cukes runner classes must run in parallel
```

## How to run?

- We can execute tests in parallel in our framework only by running tests as a maven command
- **mvn verify** command runs the tests and generate reports
- **mvn clean verify** first deletes the target folder, then runs tests, then generates reports



## JENKINS

- Jenkins
- Production Pipeline
- Continuous Integration
- Continuous Delivery
- Continuous Deployment
- Create a Job Configuration on Jenkins
- How many environment do you have?

## GIT

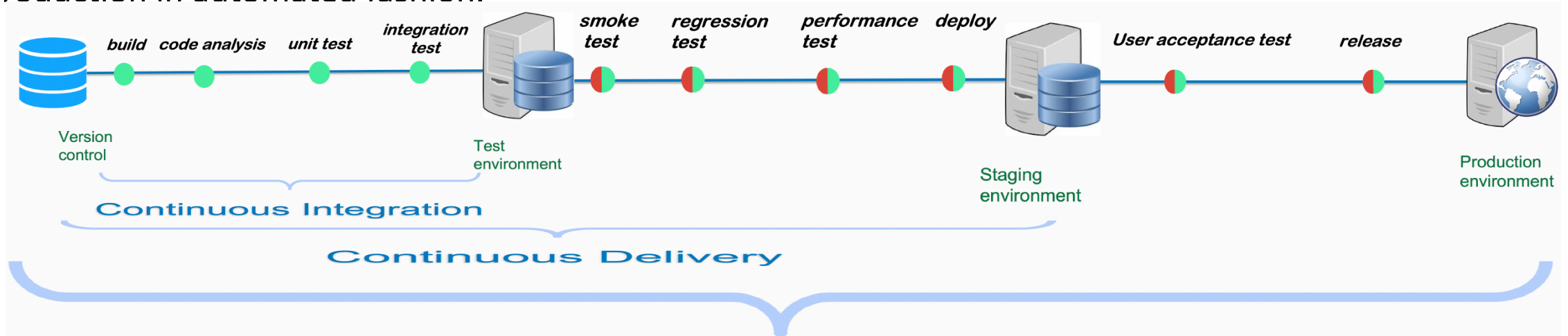
- Git Commends
- Branching Strategy
- Merge a Branch with Master Branch
- How to handle Merge Conflict

## Maven

# Jenkins



- Jenkins used to automate the processes related to deployment and delivery.
- Jenkins is application that is hosted in some server. My company uses AWS for hosting Jenkins. Our Operations (DevOps) team set up the AWS instance, install Jenkins and other required tools.
- Pipeline is a set of processes that take the code from version control and compile, build, test and deploy to production in automated fashion.



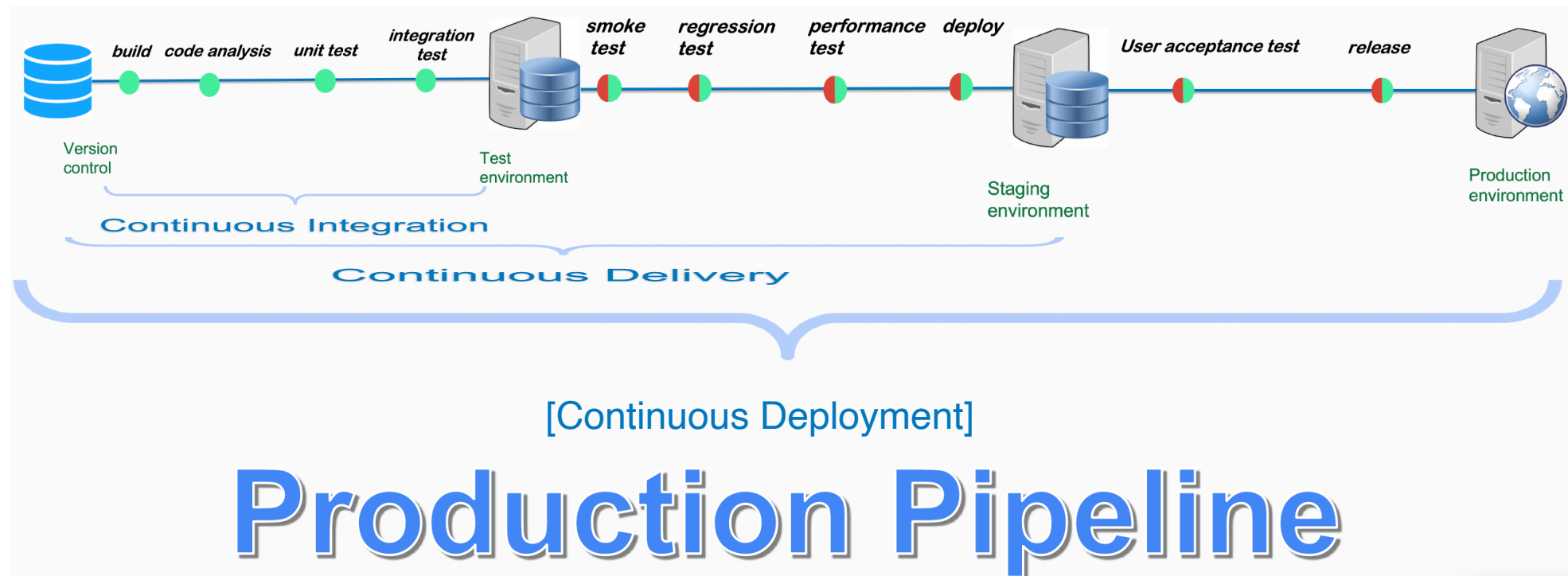
[Continuous Deployment]

# Production Pipeline

# Production Pipeline



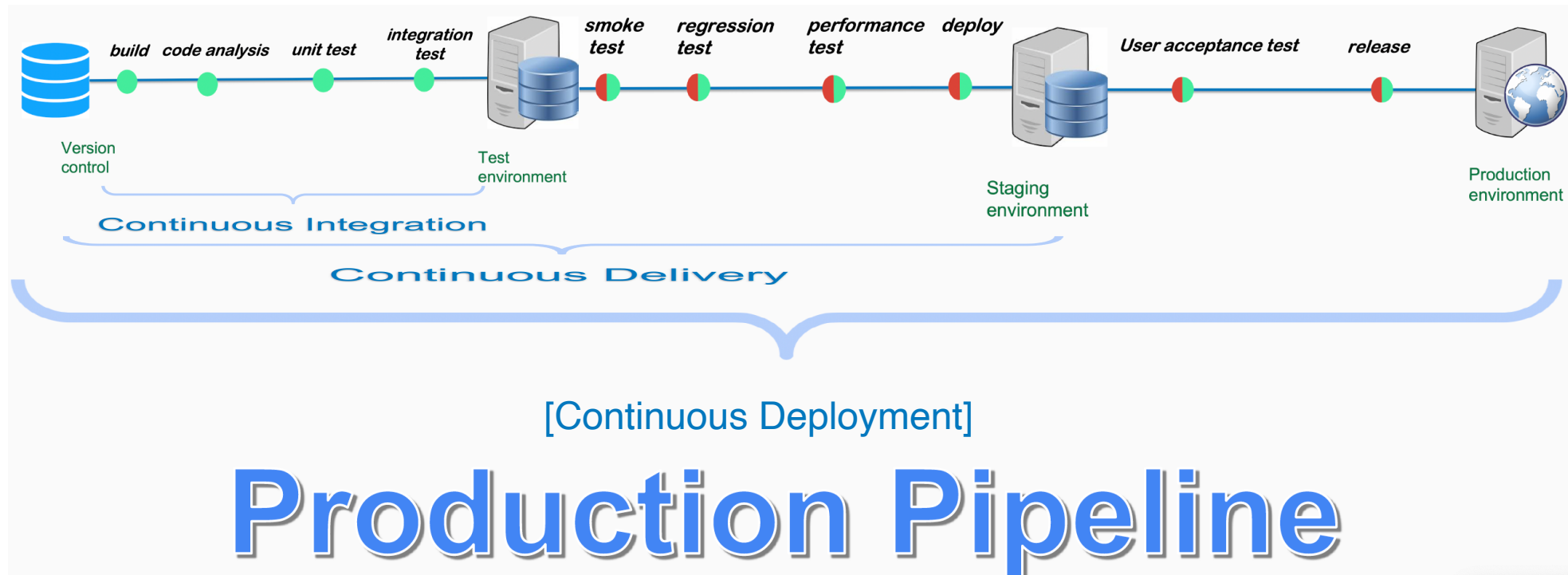
- Pipeline is a set of processes that take the code from version control and compile, build, test and deploy to production in automated fashion.
- The pipeline breaks down the software delivery process into stages. Each stage is made of different tasks which can be carried out in parallel. When all tasks in a stage passes, next stage is triggered.



# Continuous Integration



- Continuous Integration is an automated build and execution of at unit and integration tests, performing code analysis.
- The Continuous Integration process is comprised of automatic tools that assert the new code's correctness before integration. It reduces integration problems allowing to deliver software more rapidly by providing quick feedback every time new code is added to the source control. Usually Continuous Integration does not involve testing the functionality of the application.

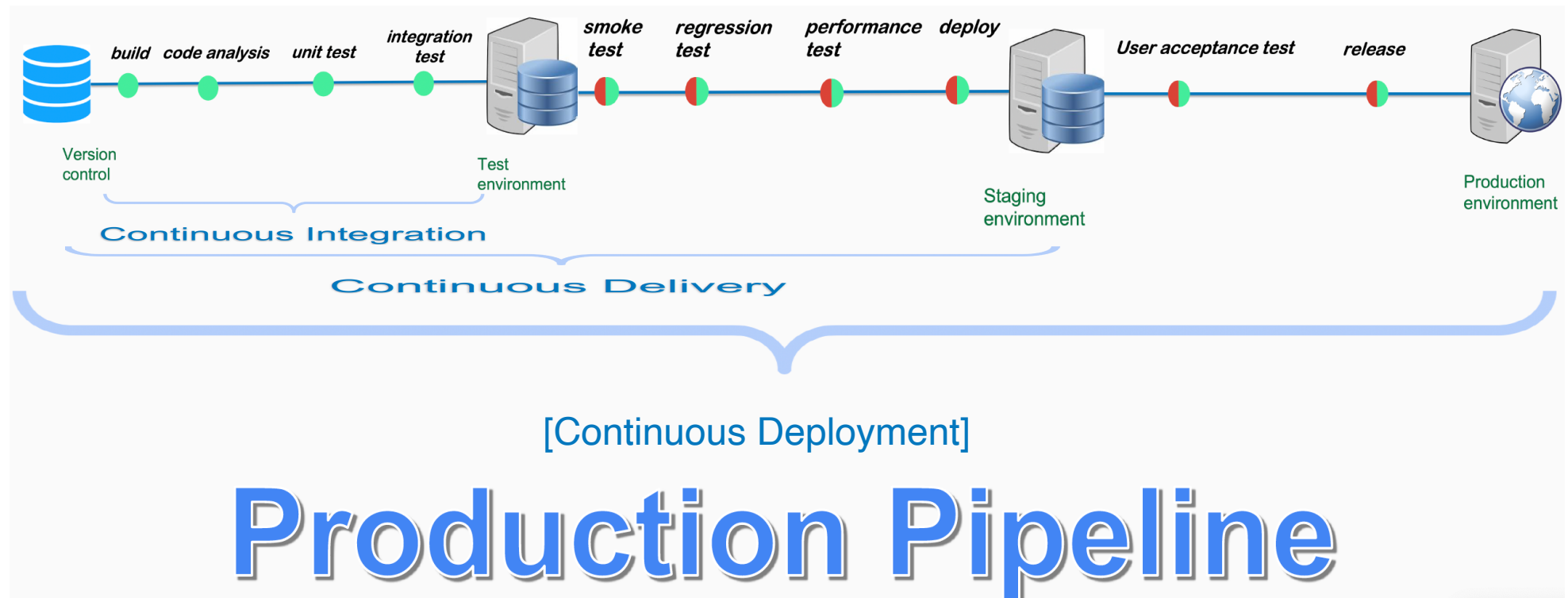




# Continuous Delivery



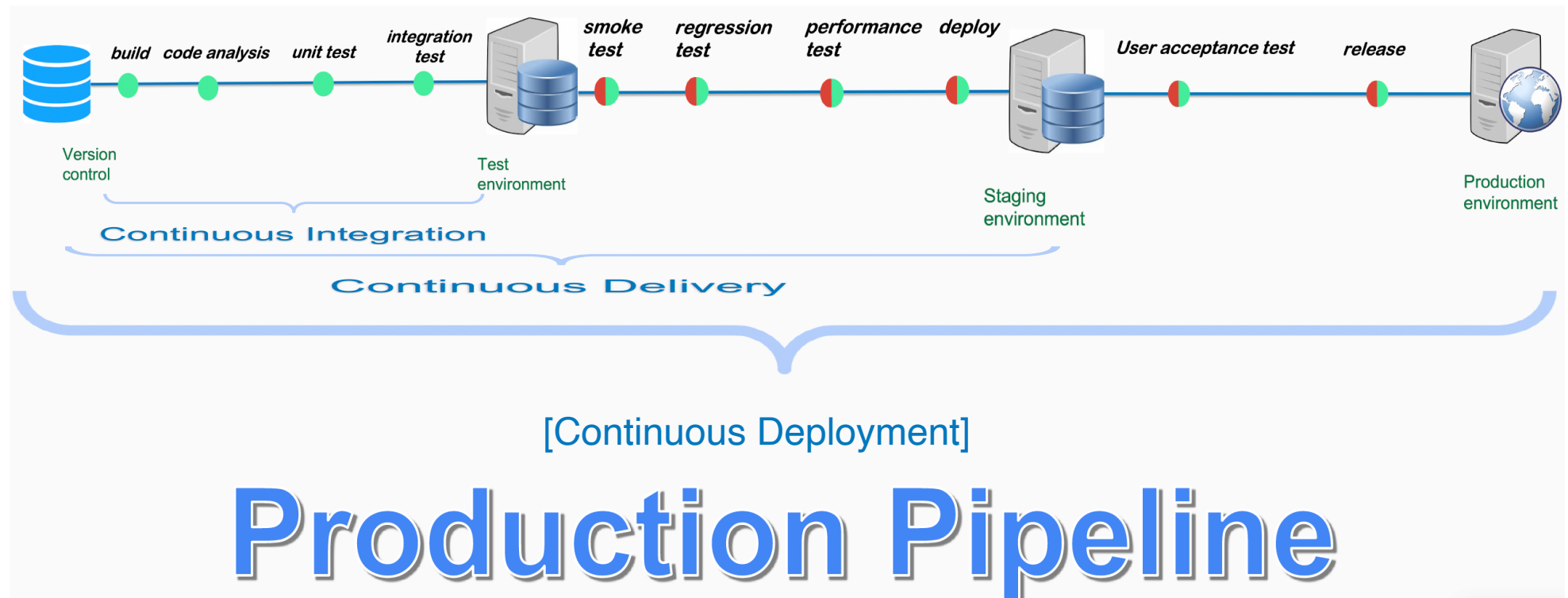
- Continuous Delivery is an automated build and execution of at unit and integration tests, performing code analysis, functional tests and also deploying to any supported platform any time. Each time a build or a set of code passes the tests, it's automatically deployed out to a staging environment. In Continuous Delivery releasing to end users is a manual process. Continuous delivery involves human decision-making when it comes to deciding when to release the software to the customers.



# Continuous Deployment



- Continuous deployment means that every change that you make, goes through the pipeline, and if it passes all the tests, it automatically gets deployed into production.
- When a developer checks in code, the automated processes take the code and move it through the entire lifecycle and if it passes each gate, it gets deployed directly to production. The delivery speeds are notably faster due to elimination of manual steps.



# Create a Job Configuration on Jenkins?



As a test engineer I only configured my smoke tests on Jenkins. I have a cucumber + junit+ maven framework. First I click the New Item link and I write the name of the test, such as smoke-test.

## 1. Source Code Management Section

here we specify where to get the code from. we put the link to our GitHub repo and also enter the credentials

## 2. Build Triggers

I specify how often I run those tests. I choose **Build periodically** because I want to run in certain schedule. In my project, I run smoke tests every morning at 6 am. So in the build trigger I entered daily option:

**H 6 \* \* \* --> every day 6 in the morning**

## 3. Build Section

Here I enter the details of the actual run. Since my project is based on maven, I choose option: **invoke top-level maven targets** then I choose which maven to run from the version dropdown. In the next field, I enter the maven goal: test. In this field we do not need to enter the word mvn. I also mention here which tag I want to run. So the command will be:

**test -Dcucumber.options="--tags @smoke"**

## 4. Add Post-build Actions

In the post build actions, I configure what I want to do after the ends. After each test I generate report and email to my team members.

For Report, I select **cucumber reports plugin** from the post-build actions to generate reports.

For email, I select **Editable Email Notification option** from the Post-build Actions to send emails to my team.

# How many environment do you have?



- 1. Dev enviroments
- 2. QA/Test enviroment- this is where I test
- 3. Staging enviroment
- 4. Production

1.Dev Environments → developer work

2.QA Environments → SDET and manual tester work

3. Stating/ pre-production Environments → regression test

4. Production Environments → real user use

# Git Commends



- **...or create a new repository on the command line**
- git init
- git add .
- git commit -m "first commit"
- git remote add origin <https://github.com/metinkaya1511/FinraDeck.git> (GitHub'daki adres)
- git push -u origin master
- **...push an existing repository from the command line**
- git remote add origin <https://github.com/metinkaya1511/FinraDeck.git> (github adresi)
- git push -u origin master

## CREATE BRANCH

- git branch develop ==> it creates new branch named 'develop' but still keep being on master branch
- git checkout develop ==> it will change your branch to the develop branch
- git checkout -b develop ==> it creates also a branch named develop and switches to it automatically

## DELETE

- git branch -d <branch\_name> **deletes** the branch. If we have unmerged changes, this command gives a warning and does not delete.
- git branch -D <branch\_name> deletes the branch even if it has unmerged changes. Gives no warning.

## SWITCH to Branch

- git checkout develop checks out the branch, switches to the branch.
- git checkout -b <branch\_name> creates a new branch and switches to it.
- git merge <branch\_name> this command takes changes from the given branch, and merges with the current branches we are on.

# Branching Strategy



- There is master branch and separate branches for each team member. when someone finishes work, they push to their own branch, then after reviewing it is merged to master.

- **HOW WE DID IT?**

- In my project we had master, develop and branch for person. so, if we have 2 automation testers, we will have:

- Master
- Develop
- Tester1
- Tester2
- Tester3

- Each tester checks in to their own branches. Then after reviewing it is merged to develop branch. We merge master and develop only once a sprint.
- In my project, our code is separate repo from the application code repo. Automation framework have a smaller code base and fewer people involved. So, we can have less complicated branching policy.

**My daily automated smoke from Jenkins runs against the master branch.** Master branch is stable since we only merge to with once a sprint.

# How to Merge a Branch with Master Branch

- - First we have to come in the branch which we want to merge the codes in. It means generally we should come into master branch in this case.
- - `git checkout master` ==> now you are in master branch
- - `git pull origin master` ==> We are pulling recent code from master branch on GitHub
- - `git merge develop -m "your message here"` ==> to merge a develop branch into master branch
- - `git add .`
- - `git commit -m "final commit"`
- - `git push origin master`
- - now when other team members pull master they will see what you sent
- \*\*\* `git rebase LoginFeatureBranch` ==> This will merge Login with Master but closes the LoginFeatureBranch for good (completely).

# How to handle Merge Conflict



1. `git stash` -- > take my project to temp memory
2. `git pull` -- > pull the project from GitHub to working directory (my computer)
3. `git stash pop` -- > take my project to my working directory,  
fix the conflict and merge the project.
4. `git add .`
5. `git commit -m "comment"`
6. `git push`





- MAVEN is a project build tool for java projects. (There are other build tools for java such as Gradle and Ant)
- MAVEN automates the process of creating, managing dependencies, compiling, testing, deploying java applications.
- Pom.XML file always located on home folder of the project. Maven automates the build process of Java projects. Each phase in the build process is known as a Maven lifecycle or a Maven goal.
- Maven is also responsible for dependencies. We have to download and add all .jar files we need for project if don't use Maven. Maven automatically downloads and adds into project. Dependencies helps us to easily add libraries and make project independent from IDE. We can run tests without IDE. (Jenkins execute tests by using Maven, not any IDE)

- **Maven Default Lifecycles:**

- ☐ **clean** - remove all files generated by the previous build (deletes Target file)
- ☐ **validate** - validate the project is correct and all necessary information is available
- ☐ **compile** - compile the source code of the project
- ☐ **test** - test the compiled source code using a suitable unit testing framework
- ☐ **verify** - run any checks to verify the package is valid and meets quality criteria
- ☐ **package** - converts built project into distributable version such as .jar or .war
- ☐ **install** - Install the built artifact into the local repository.
- ☐ **deploy** - Deploy the built artifact to the remote repository.



## AWS

- AWS – EC2 and S3
- AWS – IAM – RDS – Lambda
- AWS – Cloud Formation-AutoScaling Group –Load Balancer
- AWS - Cloud Deployment Model
- AWS – Type of Cloud Computing

## JMeter

## Linux

# AWS – EC2 and S3



- In terms of AWS capability, we use in our company:
- **EC2:**
  - for Running application for different environments(Test, Dev environments).
  - and also for Running Jenkins or any other application for testing purpose.
  - and to Run load testing on the application.
- **We also use S3 (Simple Storage Service):**
  - It is Used for file storage, test data storage.
  - It Can be used for part of data processing workflow.
  - and also it Can also be used as code repository.

Continue ->

# AWS – IAM – RDS - Lambda



- **IAM** (Identity and Access Management) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
  - I use IAM for Permission control of users, applications, typically not configured by application team but by a cloud team.
- **RDS** (Create a sample Postgresql database and connect using Dbeaver):
  - is Used for database, launch database for different environments to support the application.
- **Lambda**:
  - Serverless service that can provide quick and easy solution to run your code without the involvement of EC2.
  - Typically used for data processing incoming files.

Continue ->



- **CloudFormation:**
  - Used for launching application servers and all related AWS services. It is also usually called infrastructure as code.
- **Auto scaling group:**
  - Used for scaling servers based on the server load.
- **Load Balancer:**
  - Used for balancing incoming traffic across multiple servers.

Continue ->

# AWS - Cloud Deployment Model



## Cloud Deployment Model



### PUBLIC CLOUD

everything is the cloud or migrated to the cloud



### HYBRID

combination of on-premise and cloud



### ON-PREMISES

everything is in your own data center,

## Cloud Deployment Model

A cloud-based application is fully deployed in the cloud and all parts of the application run in the cloud. Applications in the cloud have either been created in the cloud or have been migrated from an existing infrastructure to take advantage of the benefits of cloud computing.



## Cloud Deployment Model

A hybrid deployment is a way to connect infrastructure and applications between cloud-based resources and existing resources that are not located in the cloud. The most common method of hybrid deployment is between the cloud and existing on-premises infrastructure to extend, and grow, an organization's infrastructure into the cloud while connecting cloud resources to internal system

## Cloud Deployment Model

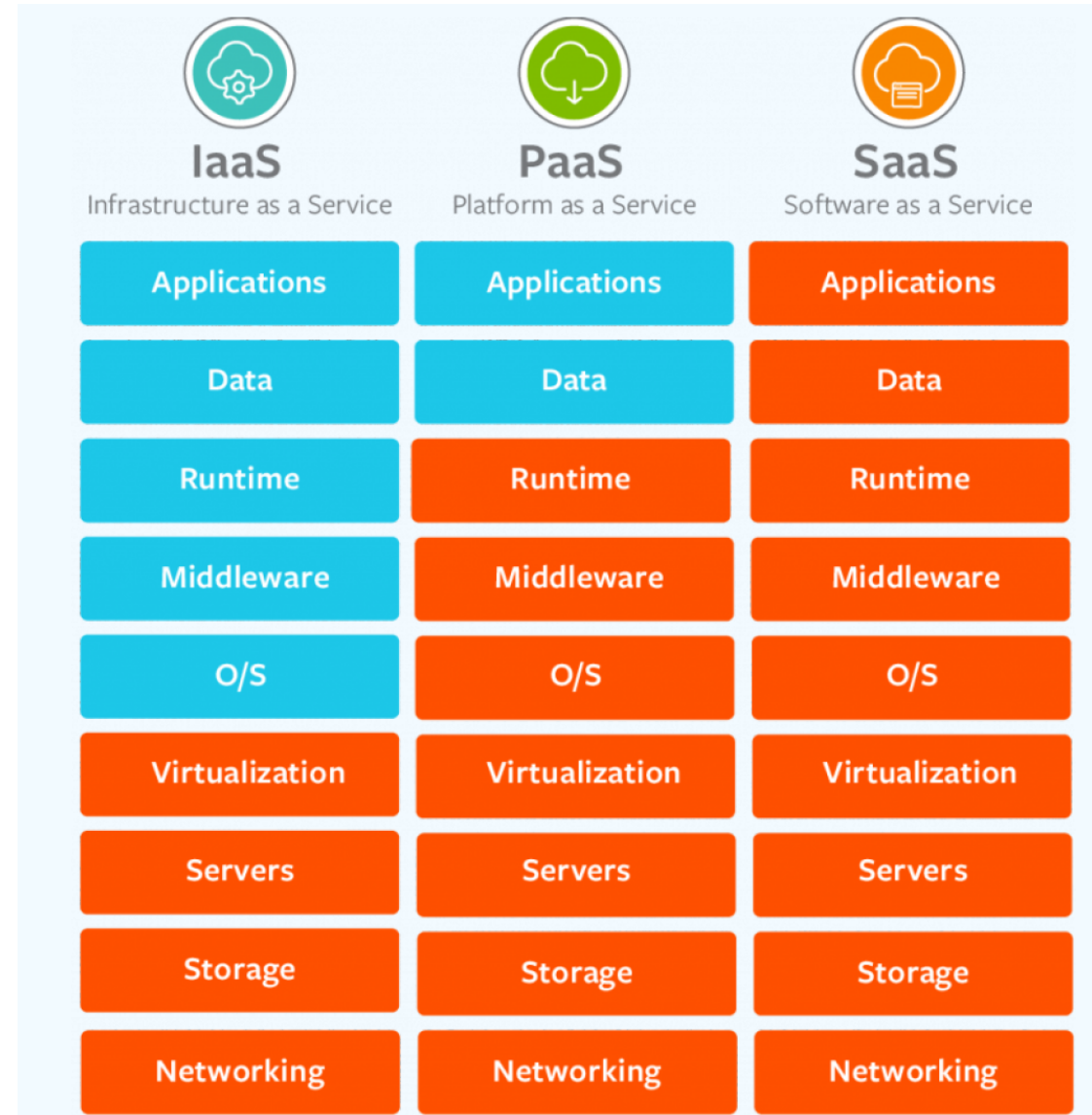
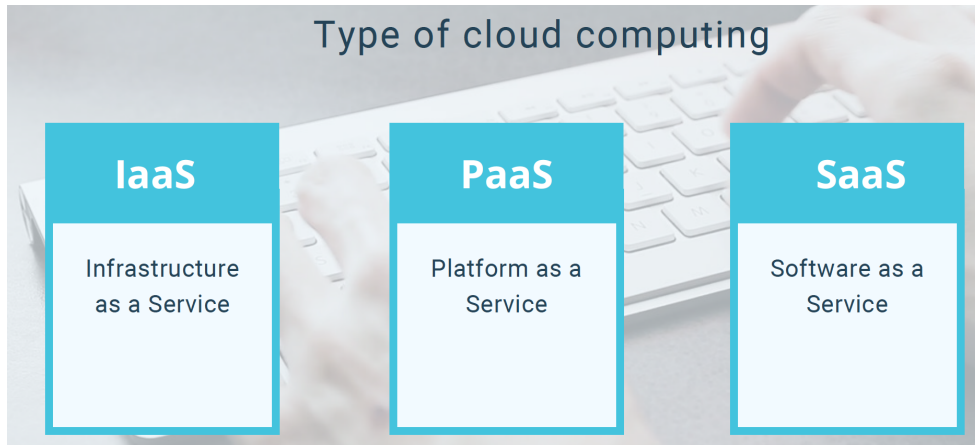


Deploying resources on-premises, using virtualization and resource management tools, is sometimes called "private cloud". On-premises deployment does not provide many of the benefits of cloud computing but is sometimes sought for its ability to provide dedicated resources.

# AWS – Type of Cloud Computing



Type of cloud computing



# Linux Commands



- Linux Commands (case-sensitive)
- reboot ==> reboots system
- man ==> gives you instruction of the command - Ex:
  - "man reboot"
- mkdir ==> Creates directory(folder)
- cd ==> Change directory
- ls ==> List directory content
- pwd ==> Print name of the current working directory.  
It
- gives you the exact location; Ex: /home/Andy/Desktop
- ll ==> Long list format
- ls-la ==> Prints files and hidden file
- clear ==> Clear screen
- cd.. ==> Goes to the parent file (not the root file)
- cd/ ==> Goes to the parent root file
- cd~ ==> Goes to the home of the user file
- grep ==> Prints a line matching a pattern
- df-h ==> Prints the disk space usage top ==> Displays
- linux tasks (like task manager)
- **How to create an account:**
  - User ==> useradd OK
  - Group ==> groupadd groupName
- **Adding a user into group:**
  - useradd -G groupName OK
  - id Andy prints details for this individual (shows it Andy has smth...)





- JMeter is a software that can perform load test, performance-oriented business (functional) test, regression test, etc., on different protocols or technologies.
- JMeter is a Java desktop application with a graphical interface that uses the Swing graphical API. It can therefore run on any environment / workstation that accepts a Java virtual machine, for example – Windows, Linux, Mac, etc.

## The tests that we can execute via the JMeter:

- **Performance Test** – This test sets the best possible performance expectation under a given configuration of infrastructure. It also highlights early in the testing process if any changes need to be made before the application goes into production.
- **Load Test** – This test is basically used for testing the system under the top load it was designed to operate under.
- **Stress Test** – This test is an attempt to break the system by overwhelming its resources.



# Java Technical Questions

N796472229

- Even-Odd
- Swap Numbers
- Reverse String
- Reverse Integer
- Prime Numbers
- String Palindrome
- Integer Palindrome
- Factorial
- Sum of Digits
- Fibonacci
- Find Duplicate Numbers

- FizzBuzz
  - Only Unique Letters
  - Remove Duplicates
  - Largest Number in Array
  - Armstrong Number
  - Diamond of Asterisks
  - Pyramid Of Numbers
  - Floyds Triangle
  - Anagram
- Count String and Letters

# Even - Odd



```
int num = put any number;
if(num % 2 == 0) {
 System.out.println(num + " is even");
} else {
 System.out.println(num + " is odd");
}
```

# Swap Numbers

- $x = x + y;$   
 $y = x - y;$   
 $x = x - y;$

# Duplicate Numbers



```
public class DuplicateNumbersCount { // will print duplicate numbers
```

```
 public static void main(String[] args) {
```

```
 int[] arr = { 1, 35, 2, 3, 5, 2, 2, 3 };
```

```
 int count = 0;
```

```
 for (int j = 0; j < arr.length; j++) {
```

```
 for (int k = j + 1; k < arr.length; k++) {
```

```
 if (arr[j] == arr[k]) {
```

```
 count++;
```

```
 }
```

```
 }
```

```
 if (count == 1)
```

```
 System.out.println(arr[j]);
```

```
 count = 0;
```

```
 }
```

```
}
```

```
}
```

## Count String and Letters

```
//Count words
```

```
public static void main(String[] args) {
String str = "This Java Casses Are Very Interesting interseting Very Java Are not in the
class";
```

```
String[] allWords= str.split(" ");
```

```
HashMap<String, Integer> wordCount = new HashMap<>();
```

```
for (int i=0; i<allWords.length; i++) {
if (wordCount.containsKey(allWords[i])) {
int temp = wordCount.get(allWords[i]); // get getting value only per Vall
wordCount.put(allWords[i], ++temp);
}
else {
wordCount.put(allWords[i], 1);
}
}
System.out.println(wordCount);
```

```
+++++
```

```
// Count letters
```

```
public class CountStringsWithHashMap {
```

```
public static void main(String[] args) {
```

```
String str = "This Java Casses Are Very Interesting";
```

```
str=str.replace(" ", "").strip().toLowerCase();
```

```
HashMap<Character, Integer> letterCount = new HashMap<>();
```

```
for (int i=0; i<str.length(); i++) {
if (letterCount.containsKey(str.charAt(i))) {
int temp = letterCount.get(str.charAt(i)); // get getting value only per Vall
letterCount.put(str.charAt(i), ++temp);
}
else {
letterCount.put(str.charAt(i), 1);
}
}
System.out.println(letterCount);
```

# Reverse String



```
public static void main (String[] args) {
 String word = "";
 for(int i = word.length()-1; i >=0; i--) {
 System.out.print(word.charAt(i))
 }
}
```

```
reverse += str.charAt(i);
return reverse; can be used in method
```

# Reverse Integer



- ```
int num = 1234, reversed = 0;
while(num != 0) {
    int digit = num % 10;
    reversed = reversed * 10 + digit;
    num /= 10; }
System.out.println("Reversed Number: " + reversed); }}
```

Prime Numbers



- `public static boolean isPrime (int number) {`
- `for(int i=2; i < number ; i++) { if(number%i == 0) {`
`return false; }}`
`return true; }`

String Palindrome



- `for(int i=word.length()-1;i>=0;i--) {`
- `reverse+=word.charAt(i); }`
- `if(word.equalsIgnoreCase(reverse)) {`
- `System.out.println("The word is palindrome"); }else {`
- `System.out.println("The word is not palindrome"); }`

Integer Palindrome



- `public static boolean isPalindrome(int number) {`
- `int palindrome = number;`
`int reverse=0;`
`while (palindrome != 0) {`
`int remainder=palindrome%10;`
`reverse = reverse*10 + remainder;`
`palindrome = palindrome/10;`
`if (number == reverse) {`
`return true; }`
`return false;`
- `}} }`

Factorial



- ```
int number = 10;
int factorialSum = 1;
for(int i = 1 ; i <=number; i++) {
 factorialSum = factorialSum * i; }

System.out.println("Factorial of " + number + " is " + factorialSum); }
```

# Sum of Digits



```
int number = 1346;
```

```
int sum = 0;
```

```
while(number > 0) {
```

```
 sum += number%10;
```

```
 number = number/10;
```

```
}
```

```
System.out.println(sum);
```

```
}
```

# Fibonacci



```
public class FabinachiNumbers {

 public static void main(String[] args) {

 int num1=0; // first number is 0

 int num2=1; // 2nd number is 1

 int max=10; // max range of given number

 System.out.println("First "+max+" Fibonacci numbers");

 for(int i=0;i<max;i++) {

 System.out.print(num1+" ");

 int sum=num1 + num2; // sum will give us next number in the line

 num1 = num2; // 1 , 1, 2, 3, 5, 8, 13, 21

 num2=sum; //1 , 2, 3, 5, 8, 13, 21, 34

 }
 }
}
```

# FizzBuzz



```
public static void main(String[] args) {
 for(int i = 1 ; i<=50 ; i++) {

 if (i % (5*3) == 0) {
 System.out.println("FizzBuzz");
 } else if(i%3==0){
 System.out.println("Fizz");
 }else if(i%5==0) {
 System.out.println("Buzz");
 }else {
 System.out.println(" " + i);
 }
 }
}
```

# Only Unique Letters



```
public static String uniqueChars(String str) {
 String unique = "";
 for (int i = 0; i < str.length(); i++) {
 if (!unique.contains("" + str.charAt(i))) {
 unique += str.charAt(i);
 }
 }
 return unique;
}
```

# Remove Duplicates



- converting the ArrayList to a HashSet effectively removes duplicates
- `Set<String> s = new LinkedHashSet<>(list);`



# Largest Number in Array



- `int [] arr = {5, 6, 76, 31, 43, 1};`
- `Arrays.sort(arr);`
- `System.out.println (arr[ arr.length - 1 ]);`

# Armstrong Number



- `public static void main(String[] args) { Scanner scan=new Scanner(System.in);  
System.out.println("Please enter number"); int number=scan.nextInt();  
System.out.println("Please enter number of digits");`
- `int digit=scan.nextInt(); int temp=number;  
int sum=0;  
do {`
- `int value = temp % 10;  
temp /= 10;  
sum += Math.pow (value , digit);  
} while (temp>0);  
if (number==sum) {  
System.out.println("This is an Armstrong number");  
}else {  
System.out.println("This is not an Armstrong  
number"); }}`

# Diamond of Asterisks



# Pyramid Of Numbers



- Scanner scan=new Scanner (System.in);
- System.out.println("How many rows you want in your pyramid?");  
int noOfRows=scan.nextInt();  
int rowCount=1;  
System.out.println("Here is your pyramid");  
for(int i=noOfRows;i>0;i--) {  
for(int j=1;j<=i;j++) {  
System.out.print(" "); }  
for(int j=1;j<=rowCount;j++) { System.out.print(rowCount+ " "); }  
System.out.println();  
rowCount++; }}}



# Floyds Triangle

- Floyd's triangle is the right angled triangle consists of natural numbers in the following fashion.  
Floyd's Triangle :
- ```
1
23
456
7 8 9 10
11 12 13 14 15
```
- ```
public static void main(String[] args) {
```
- ```
System.out.println("How many rows you want in Floyd's Triangle?");
Scanner scan = new Scanner(System.in);
int noOfRows = scan.nextInt();
```
- ```
int value = 1; System.out.println("Floyd's Triangle : "); for(int i=1 ; i<=noOfRows ;
i++) {
for(int j=1 ; j<=i ; j++) { System.out.print(value+ " ");
value++; }
System.out.println(); }}
```



# Anagram

What Is Anagram? Two strings are called anagrams if they contain same set of characters but in different order. For example, “Dormitory – Dirty Room”, “keep – peek”, are some anagrams.

```
public static void main(String[] args) {

 String word1 = "abaf ";
 String word2 = "baa";

 isAnagram(word1, word2);
}

static void isAnagram(String s1, String s2) {

 // Removing all white spaces from s1 and s2 String copyOfs1

 String copyOfs1 = s1.replaceAll("\\s", "");
 String copyOfs2 = s2.replaceAll("\\s", "");

 boolean status = true;

 if (copyOfs1.length() != copyOfs2.length()) {

 status = false;
 }
}
```

continuation

```
else {
 char[] s1Array = copyOfs1.toLowerCase().toCharArray();
 char[] s2Array = copyOfs2.toLowerCase().toCharArray();

 Arrays.sort(s1Array);
 Arrays.sort(s2Array);
 status = Arrays.equals(s1Array, s2Array);
}
if (status) {
 System.out.println(s1 + " and " + s2 + " are anagrams");
} else {

 System.out.println("'" + s1 + " and " + s2 + " are not anagrams");
}
}
```

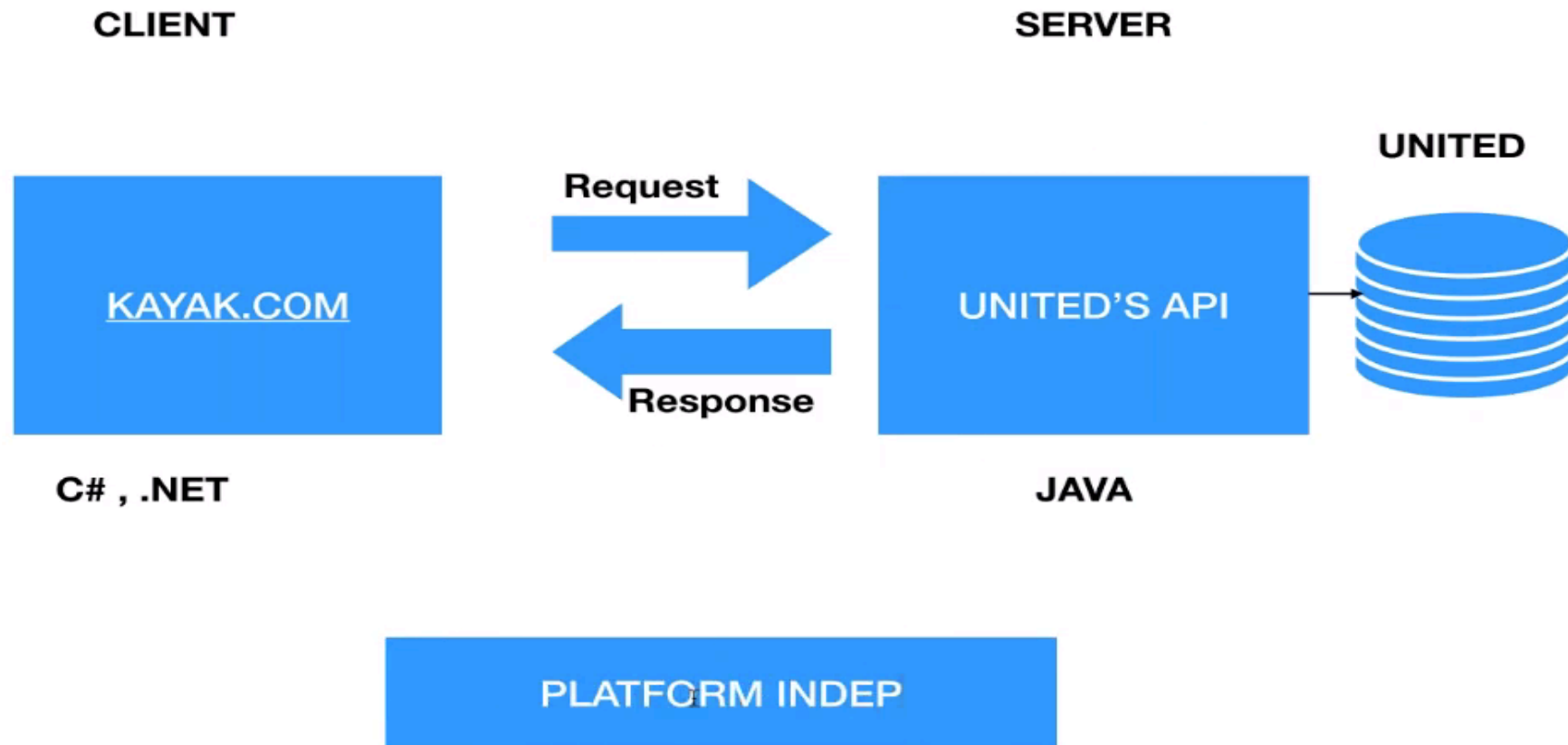
# API

What is the API?	What Response includes?	Authentication VS Authorization
HOW DO WE TEST APIS	API TEST STRATEGY	HOW TO CONVERT JSON TO JAVA OBJECT?
What are the Http methods and request types	How do you test rest api?	DE-SERIALIZATION / SERIALIZATION
API Documentation - Swagger	Ways to navigate JSON and VERIFYING RESPONSE DATA	Can All API endpoints use all of the Http protocols?
Do you have API documentation website for your API?	VERIFYING RESPONSE DATA:	Difference between SOAP and RESTful web services?
PARAMETERS IN REST SERVICES	When I do negative testing	End to End Testing Scenarios: UI, API, DB
When to use @QueryParam vs @PathParam	What first thing you check when you get response?	TYPES OF BUGS WE FACE WHEN PERFORMING API TESTING
How do you test rest api?	API STATUS CODES - HTTPS STATUS CODE	CHALLENGES IN API TESTING
What are headers in REST API?	What methods are you using to verify the size of the response data?	Test flows
What Request Line includes?	Why do you use JSONPath?	My API testing role in my current project



## API -> APPLICATION PROGRAMMING INTERFACE

CLIENT -> -> -> SERVER(API SERVER)  
CLIENT <- <- <- SERVER(API SERVER)



API is a middle man between database and client.





# API Interview Questions

- If API communication happens through internet , we can also call it **web service**.
- **HOW DO WE TEST APIs?**
- As we know in API, there is request and response communication happens between client and server.
- As testers, we send a request to an API and verify the response.
- Request -> types of requests in RestApi:
  - > GET request -> READ data (like SELECT in sql)
  - > POST request -> is to CREATE data
  - > PUT request -> UPDATE data
  - > DELETE request -> DELETE data

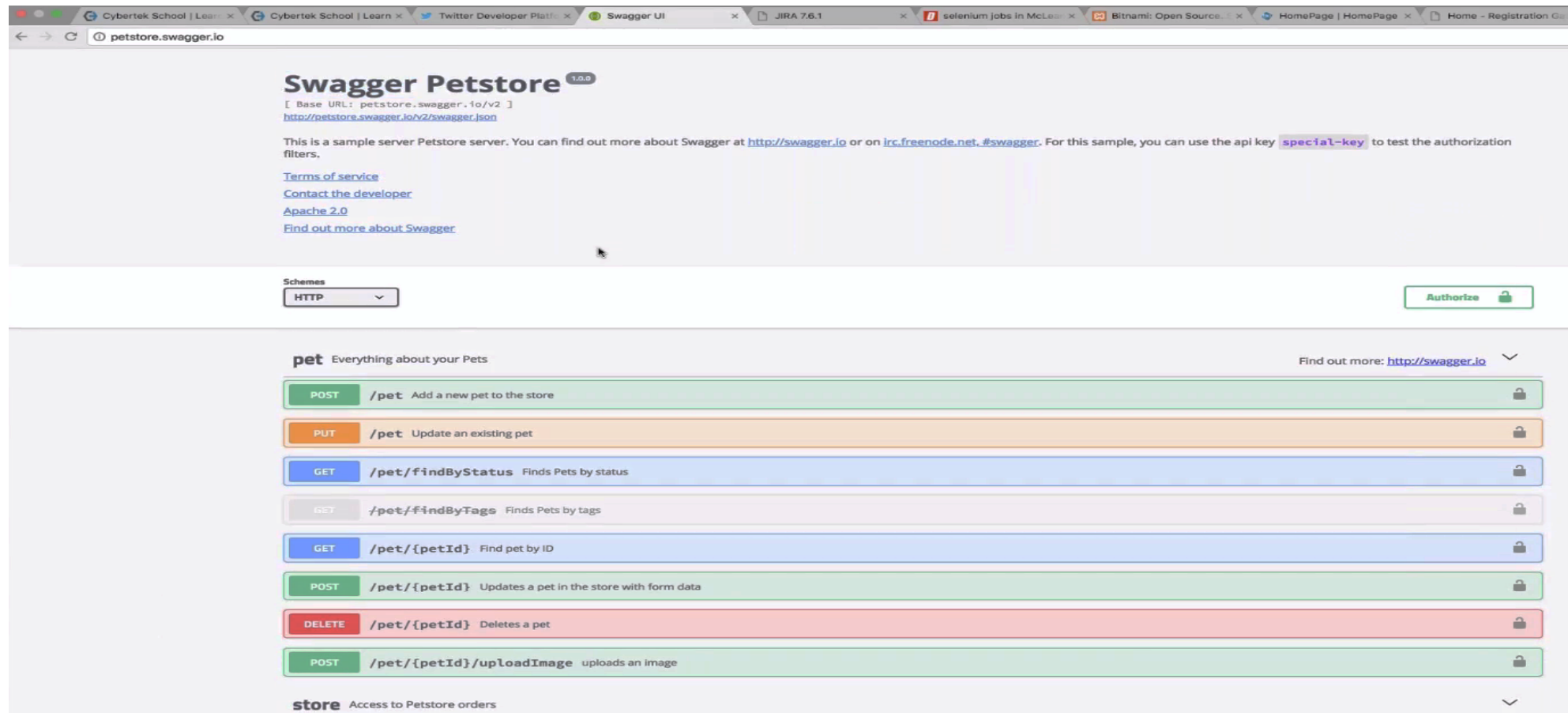


# What are the **Http methods** and **request types**

- **Get** does not requires body
  - (retrieves data from given server using a given URI)
- **Put** requires body means UPDATE information
  - (Replaces all current representations of the target resource with the uploaded content)
- **Post** requires body means CREATE information
  - (send data to the server)
- **Delete** does not requires body
  - (Removes all current representations of the target resource given by a URI.)

# API Documentation - Swagger is a tool for API documentation.

- **Swagger** is a tool for API documentation.
- API documentation is a technical content deliverable, containing **instructions about how to effectively use and integrate with an API**. It's a concise reference manual containing all the information required to work with the API



The screenshot shows the Swagger Petstore API documentation interface in a web browser. The browser's address bar displays `petstore.swagger.io`. The page header includes the title "Swagger Petstore" with a version indicator "1.0.0". Below the title, it provides the base URL `petstore.swagger.io/v2` and a link to the Swagger JSON file `http://petstore.swagger.io/v2/swagger.json`. A descriptive paragraph states: "This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on [#swagger](https://irc.freenode.net). For this sample, you can use the api key `special-key` to test the authorization filters." Below this text are links for "Terms of service", "Contact the developer", "Apache 2.0", and "Find out more about Swagger".

The interface features a "Schemes" dropdown menu set to "HTTP" and an "Authorize" button. The main content area lists the API endpoints under the "pet" category, described as "Everything about your Pets". A link to "Find out more: <http://swagger.io>" is provided. The endpoints are as follows:

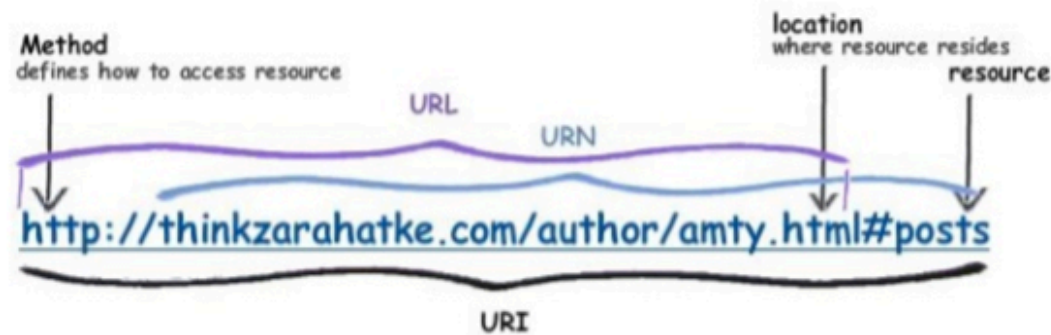
Method	Endpoint	Description
POST	<code>/pet</code>	Add a new pet to the store
PUT	<code>/pet</code>	Update an existing pet
GET	<code>/pet/findByStatus</code>	Finds Pets by status
GET	<code>/pet/findByTags</code>	Finds Pets by tags
GET	<code>/pet/{petId}</code>	Find pet by ID
POST	<code>/pet/{petId}</code>	Updates a pet in the store with form data
DELETE	<code>/pet/{petId}</code>	Deletes a pet
POST	<code>/pet/{petId}/uploadImage</code>	uploads an image

At the bottom, the "store" category is listed with the description "Access to Petstore orders".

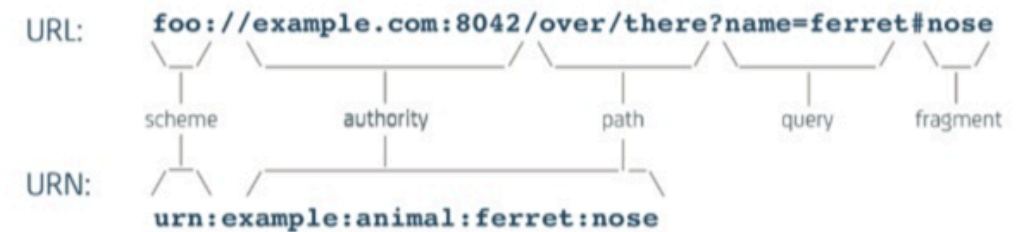


# Do you have API documentation website for your API?

- Yes we use swagger for our api documentation and this is where the description and guidelines of API endpoints are



The structure of URIs



- URL(Uniform Resource Locator) ==> `https://www.google.com/index.html`
- URN(Uniform Resource Name) ==> `www.google.com/index.html`
- URI(Uniform Resource Identifier) ==> `https://www.google.com/index.html`
- When we are doing something with API, it means that we are skipping UI and directly get the data/info from Web Services.

# 2 TYPES OF PARAMETERS IN REST SERVICES: 🏠

## 1) QUERY/REQUEST PARAMETERS

-> is not part of url and passed in key+value format  
those parameters must be defined by API developer

<http://34.223.219.142:1212/ords/hr/employees?limit=100>

## 2) PATH PARAMETERS

-> is a part of URL and followed by the end of full resource url

<http://34.223.219.142:1212/ords/hr/employees/100>

# When to use @QueryParam vs @PathParam🏠

- If there is a scenario to retrieve a record **based on id**, for example you need to get the details of the employee whose id is 15, then you can have resource with @PathParam.

GET /employee/{id}

- If there is a scenario where you need to get the **details** of all employees but only 10 at a time, you may use query param

GET /employee?**start=1&size=10**

(This says that starting employee id 1 get ten records.)

- To summarize, use @PathParam for retrieval based on id. Use @QueryParam for filter

# How do you test rest api?



- I use POSTMAN for manual API testing and use RESTASSURED library in Java for automation.
- I verify if each REST API endpoint is working as expected.
- I send POST,PUT,GET, DELETE type of requests and verify response status code and response body and header.

# What are headers in REST API?



- I am using **Accept.(ContentType.JSON)** type checks what I am **receiving** should be in **JSON** or **XML** format
- I am using **ContentType.(ContentType.JSON)** checks what I am **sending** should be in **JSON** format





# What **Request Line** includes?

- **end point** —> address where we send the request

- **base url** where API is

- **resources** resources inside baseURL

- **parameters** separated from resources with '?'.

**GET** —> get some data without changing it in the server.

- accept type: response in Json or xml

- authorization tokens, credentials

**POST** —> add data to the server. Parameters sent in xml/json as part of request body/payload.

- accept type: response in Json or xml

- content type: request body in Json or xml

- authorization tokens, credentials

**PUT** —> replaces existing data in the server.

**DELETE** —> deletes data.

*// APIs do different types of operations: **Create Read Update Delete**.*



# What **Response** includes?

- **Status code** —> defines in the request was successful.
- **Header** —> metadata (Time of execution, size etc.)
- **Body** —> returned information from the server. responses can be in different format (Json, XML, text, HTML)

# API TEST STRATAGY



- API testing involves APIs directly and checks whether the API meets expectations in terms of **functionality, reliability, performance, and security** of an application. My first concern is **functional testing** which ensures that the API functions correctly.
- The main objectives in **functional testing** of the API are:
  - to ensure that the implementation is working correctly as expected - no bugs!
  - to ensure that the implementation is working as specified according to API documentation.
  - to prevent regressions between code merges and releases.



# I HAVE FOUR DIFFERENT PROCESS TO IMPLEMENT

## • Checking API contract –SWAGER:

An API is essentially a contract between the client and the server or between two applications. Before any implementation test can begin, it is important to make sure that the contract is correct.

- Endpoints are correct,
- Resource correctly reflects the object model (proper JSON/XML structure used in response),
- There is no missing functionality or duplicate functionality,
- Relationships between resources are reflected in the API correctly.
- Now, that we have verified the API contract, we are ready to think of what and how to test.

## • Creating test cases

- I mostly create :
- Basic positive test (happy paths)
- Extended positive testing with optional parameters and extra functionality.
- Negative testing with valid input (trying to add an existing username)
- Negative testing with invalid input (trying to add a username which is null)
- Destructive testing (sending null, empty string, integer or other types, odd date format, deleting necessary parameters)
- Security, authorization, and permission tests (sending valid or invalid access tokens to permitted or unpermitted endpoints)

## • Executing test cases

- For each API request I need to verify:
- I check **Data accuracy**: I check the request and response body whether those are as written on API documentation in terms of data type and data structure.
- I check **HTTP status code**: For example, creating a resource should return 201 CREATED and unpermitted requests should return 403 FORBIDDEN, etc.
- I check **Response headers**: HTTP server headers have implications on both security and performance.
- I check **Response body**: Check valid JSON body and correct field names, types, and values - including in error responses.
- I check **Authorization checks**: Check authentication and authorization
- I check **Error messages**: Check the error code coverage in case API returns any error
- I check **Response time**: Implementation of response timeout

## • Implementing different test flows

- Single-step workflow:
- Multi
- Combined API and UI test: -step workflow with several requests:

# How do you test rest api?



- I also do positive and negative testing of API.
- **When I do positive testing,**
  - I send
    - valid request parameters ,
    - valid headers,
    - valid request JSON body and
  - verify that response status code is 200 successful and JSON response body data is also matching the expected.
- **When I do negative testing,**
  - I send
    - invalid request parameters , or
    - invalid headers, or
    - invalid request json body and
  - verify that response status code is not 200 and Json response body contains error message.

# Ways to navigate JSON and VERIFYING RESPONSE DATA



## 1) Not recommended:

treat the response json as a String and do contains assertions on it.

```
AssertTrue(response.body().asString().contains("Java"));
```

## 2) PATH() method.

Extract values from JSON using path() method, use Junit assertions for verification.

```
String city = response.path("employee.address.city");
assertEquals("New York", city);
```

## 3) JSONPATH object:

Convert Response data into JsonPath object and use jsonpath getter methods to extract values. Do assertions using JUnit

```
JsonPath json = response.jsonPath();
assertEquals("Ayse", json.getString("teachers.first_name"));
```

## 4) HAMCREST MATCHERS WITH PATH USING CHAINING:

We can do assertions in single statement by chaining methods in restAssured. To find values in the json body, we use the same path syntax: and().assertThat()

```
.body("teachers.firstName",contains("Esen"),
 "teachers.lastName",contains("Niiazov"),
 "teachers.emailAddress",contains("eniiazov@gmail.com"));
```

## 5) Java Collections/Data Structures to manipulate Json Data. JSON RESPONSE --> JAVA DATA STRUCTURE/COLLECTION

Response Json Data:

```
{
 "year", 2000,
 "make", "Honda"
 "model", "f500",
 "mileage", 50234
}
```

```
Map<String, Object> dataMap = response.body().as(Map.class);
dataMap.get("year") => 2000
dataMap.get("make") => Honda
```

```
http://api.cybertektraining.com/teacher/name/{name}
{
 "teachers": [
 {
 "teacherId": 2381,
 "firstName": "Esen",
 "lastName": "Niiazov",
 "emailAddress": "eniiazov@gmail.com",
 "joinDate": "01/01/2018",
 "password": "123456123456",
 "phone": "12345678910",
 "subject": "Java",
 "gender": "Male",
 "department": "Computer",
 "birthDate": "01/01/1992",
 "salary": 100000,
 "batch": 11,
 "section": "1",
 "premanentAddress": "2700 S. River, Des Plaines, IL 60600"
 }
]
}
```



## VERIFYING RESPONSE DATA:

1) **asString() method convert body to String.** treat the response json as a String and do contains assertions on it.

```
String responseBody = response.body().asString();
assertTrue(responseBody.contains("Java"); //assertTrue(response.body().asString().contains("Java"));
```

2) **PATH() method.** Extract values from JSON using path() method, use Junit assertions for verification.

```
String city = response.path("employee.address.city");
assertEquals("New York", city);
```

3) **JSONPATH object:** Convert Response data into JsonPath object and use jsonpath getter methods to extract values. Do assertions using JUnit.

```
json.getString("x", "y"), json.getInt(a, b)....
JsonPath json = response.jsonPath();
String tName = json.getString("teachers.first_name")
assertEquals("Ayse", tName); //assertEquals("Ayse", json.getString("teachers.first_name"));
long phone = json.getLong("teachers.phone")
assertEquals(12345678910, phone);
```

4) **HAMCREST MATCHERS WITH PATH USING CHAINING:** We can do assertions in single statement by chaining methods in RESTassured. To find values in the json body , we use the same path syntax:

```
.and().assertThat()
.body("teachers.firstName",contains("Esen"),
"teachers.lastName",contains("Niiazov"),
"teachers.emailAddress",contains("eniiazov@gmail.com"));
```

5) **Java Collections/Data Structures to manipulate Json Data.**

### JSON RESPONSE --> JAVA DATA STRUCTURE/COLLECTION

Response Json Data:

```
{
 "year", 2000,
 "make", "fiat",
 "model", "f500",
 "mileage", 50234
```

```
}Map<String, Object> dataMap = response.body().as(Map.class);dataMap.get("year") => 2000
```



# When I do negative testing

```
5 /*
6 * When I send a GET request to REST URL:
7 * http://34.223.219.142:1212/ords/hr/employees/1234
8 * Then status code is 404
9 * And Response body error message is "Not Found"
10 *
11 */
12 @Test
13 public void negativeGet() {
14 // when().get("http://34.223.219.142:1212/ords/hr/employees/1234")
15 // .then().statusCode(404);
16 Response response = when().get("http://34.223.219.142:1212/ords/hr/employees/1234");
17 assertEquals(response.statusCode(), 404);
18 assertTrue(response.asString().contains("Not Found"));
19 }
```



# What first thing you check when you get response?

- Status code (200 always mean Ok)
- We always check the 404 means not found

# API STATUS CODES -HTTPS STATUS CODE



This page is created from HTTP status code information found at [ietf.org](#) and [Wikipedia](#). Click on the **category heading** or the **status code** link to read more.

## 1xx Informational

100 Continue

101 Switching Protocols

102 Processing (WebDAV)

## 2xx Success

- ★ 200 OK
- 203 Non-Authoritative Information
- 206 Partial Content
- 226 IM Used

- ★ 201 Created
- ★ 204 No Content
- 207 Multi-Status (WebDAV)

- 202 Accepted
- 205 Reset Content
- 208 Already Reported (WebDAV)

## 3xx Redirection

- 300 Multiple Choices
- 303 See Other
- 306 (Unused)

- 301 Moved Permanently
- ★ 304 Not Modified
- 307 Temporary Redirect

- 302 Found
- 305 Use Proxy
- 308 Permanent Redirect (experimental)

## 4xx Client Error

- ★ 400 Bad Request
- ★ 403 Forbidden
- 406 Not Acceptable
- ★ 409 Conflict
- 412 Precondition Failed
- 415 Unsupported Media Type
- 418 I'm a teapot (RFC 2324)
- 423 Locked (WebDAV)
- 426 Upgrade Required
- 431 Request Header Fields Too Large
- 450 Blocked by Windows Parental Controls (Microsoft)

- ★ 401 Unauthorized
- ★ 404 Not Found
- 407 Proxy Authentication Required
- 410 Gone
- 413 Request Entity Too Large
- 416 Requested Range Not Satisfiable
- 420 Enhance Your Calm (Twitter)
- 424 Failed Dependency (WebDAV)
- 428 Precondition Required
- 444 No Response (Nginx)
- 451 Unavailable For Legal Reasons

- 402 Payment Required
- 405 Method Not Allowed
- 408 Request Timeout
- 411 Length Required
- 414 Request-URI Too Long
- 417 Expectation Failed
- 422 Unprocessable Entity (WebDAV)
- 425 Reserved for WebDAV
- 429 Too Many Requests
- 449 Retry With (Microsoft)
- 499 Client Closed Request (Nginx)

## 5xx Server Error

- ★ 500 Internal Server Error
- 503 Service Unavailable
- 506 Variant Also Negotiates (Experimental)
- 509 Bandwidth Limit Exceeded (Apache)
- 598 Network read timeout error

- 501 Not Implemented
- 504 Gateway Timeout
- 507 Insufficient Storage (WebDAV)
- 510 Not Extended
- 599 Network connect timeout error

- 502 Bad Gateway
- 505 HTTP Version Not Supported
- 508 Loop Detected (WebDAV)
- 511 Network Authentication Required

Lightshot Screenshot

<b>200</b>	OK
<b>201</b>	CREATED
<b>202</b>	ACCEPTED
<b>204</b>	NO CONTENT
<b>304</b>	NOT MODIFIED
<b>400</b>	BAD REQUEST
<b>401</b>	UNAUTHORIZED

<b>403</b>	FORBIDDEN
<b>404</b>	NOT FOUND
<b>405</b>	METHOD NOT ALLOWED
<b>409</b>	CONFLICT
<b>500</b>	INTERNAL SERVER ERROR
<b>502</b>	BAD GATEWAY
<b>503</b>	SERVICE UNAVAILABLE



# What methods are you using to verify the size of the response data?

```
@Test
public void testItemsCountFromResponseBody() {
 given().accept(ContentType.JSON)
 .when().get(ConfigurationReader.getProperty("hrapp.baseresturl")+"/regions")
 .then().assertThat().statusCode(200)
 .and().assertThat().contentType(ContentType.JSON)
 .and().assertThat().body("items.region_id", hasSize(4))
 .and().assertThat().body("items.region_name", hasItem("Americas"))
 .and().assertThat().body("items.region_name", hasItems("Americas", "Asia", "Middle East and Africa"));
}
```

I use Matchers from Hamcrest

- o hasItem()
- o equalTo()



# Why do you use JSONPath?

```
APIDays_jsonPath.java configuration.properties
105 public void testWithJsonPath() {
106
107 Map<String,Integer> rParamMap = new HashMap<>();
108 rParamMap.put("limit", 100);
109
110 Response response = given().accept(ContentType.JSON)//header
111 .and().params(rParamMap) //query param/request param
112 .and().pathParams("employee_id", 177) //path param
113 .when().get(ConfigurationReader.getProperty("hrapp.baseresturl")+"/emp'
114
115 JsonPath json = response.jsonPath(); //get json body and assign to jsonPath object
116
117 System.out.println(json.getInt("employee_id"));
118 System.out.println(json.getString("last_name"));
119 System.out.println(json.getString("job_id"));
120 System.out.println(json.getInt("salary"));
121 }
```

JsonPath is used easily to navigate and manipulate JSON data

# Authentication VS Authorization



- **authentication** - *who are you?*
- **authorization** - *what rights do you have?*

# Authentication



**API KEY** = *one type authentication, we get it from the service provider (sent to account after signing up etc) and we include the key for all our requests as a parameter:*

```
given().queryParams("apikey", "a9faab96").
```

**BASIC AUTHENTICATION** = *using user name and password for authentication*

**1. Challenged basic authentication** —> *rest assured will not send username/password initially. It will only be send once server asks for it.*

```
given().auth().basic("username", "password").
```

**2. Preemptive basic authentication** —> *rest assured sends username/password before server asks for it.*

```
given().auth().preemptive().basic("username", "password").
```

# Authorization



**OAUTH Authorization** = *keys and tokens from 3rd party are used for authentication. To get token we are calling get("sign") end point. It requires username & password. API recognizes credentials and returns token.*

1. OAuth1

2. OAuth2

```
Response response = given().
 param("email", "valid_email").
 param("password", "valid_password").
 when().get(endpoint); -> to pass credentials
```

```
String accessToken = response.path().get("accessToken"); -> to get token
```

```
given().header("Authorization", token). -> to use token
```



# HOW TO CONVERT JSON TO JAVA OBJECT? 🏠

- First in first I add **GSON** dependency in to the pom.xml file

```
<dependency>
 <groupId>com.google.code.gson</groupId>
 <artifactId>gson</artifactId>
 <version>2.8.2</version>
</dependency>
```

- **GSON** -> is a json parser that is used to convert
  - from **java object** to **json** (serialization)
  - from **json** to **java object** (deserialization)

# DE-SERIALIZATION / SERIALIZATION:



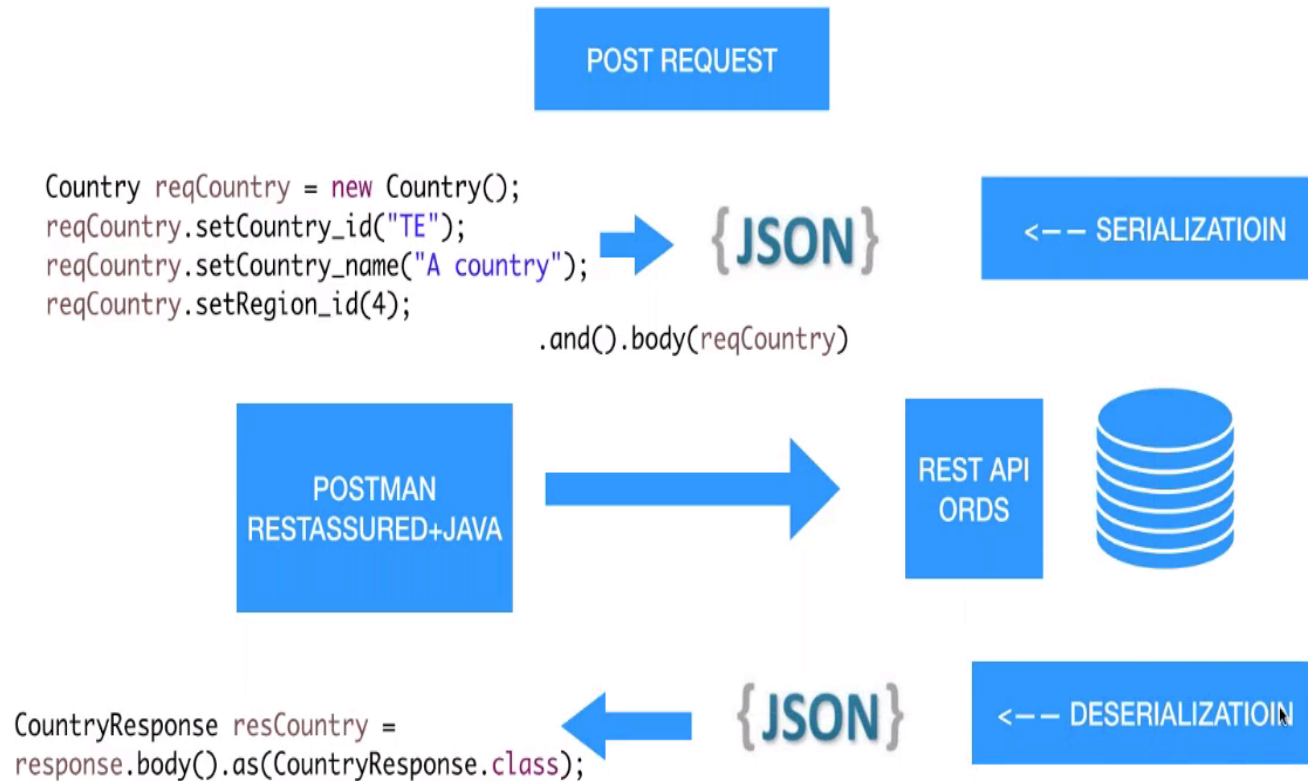
- DE-SERIALIZATION: CONVERT JSON -> JAVA OBJECT  
JSON TO MAP

```
17 public class APIDay3_GSON {
18
19 @Test
20 public void testWithJsonToHashMap() {
21 Response response = given().accept(ContentType.JSON)
22 .when().get(ConfigurationReader.getProperty("hrapp.baseresturl")+"/employees/1
23
24 Map<String,String> map = response.as(HashMap.class);
25
26 System.out.println(map.keySet());
27 System.out.println(map.values());
28
29 assertEquals(map.get("employee_id"),120);
30 assertEquals(map.get("job_id"), "AC_MGR");
31
32
```

# SERIALIZATION/ DE-SERIALIZATION



- SERIALIZATION: CONVERT JAVA OBJECT -> JSON



DE-SERIALIZATION: CONVERT JSON -> to JAVA Object



## Serialization (JAVA to JSON) and Deserialization (JSON to JAVA format)

```
@Test
public void gsonExample() {

 Gson gson = new Gson();

 //Serialization:
 Customer customer = new Customer(20, "Vlad", "male", 7033964165L);
 customer = {id=20, name='Vlad', gender='male', phone:7033964165} //java format

 String jsonCustomer = gson.toJson(customer);
 System.out.println("to json format - Serialization: " + jsonCustomer);
 //to json format-Serialization:{"id":20,"name":"Vlad","gender":"male", "phone":7033964165} //json format

 //De-Serialization:

 String myJson = "{\"id\":25,\"name\":\"Roman\",\"gender\":\"male\",\"phone\":5712223366}"; //json format

 Customer javaCustomer = gson.fromJson(myJson, Customer.class);
 System.out.println("to java format - Deserialization: " + javaCustomer.toString());

 //to java format-Deserialization: {id=25, name='Roman', gender='male', phone=5712223366} //java format

 //fromJson(String json, Which.class) --> it will convert the json to object of the class
 //toJson(java object) -> it will take the java object and create json and return it
}
```

# Can All API endpoints use all of the Http protocols?

- It depends, API developer decides if that url works with GET,POST,PUT, or DELETE requests

# Difference between SOAP and RESTful web services?

- RESTful supports JSON, XML,
- SOAP supports only XML
- REST is faster than SOAP based web services
- SOAP is more secure
- REST is getting more popular



# End to End Testing Scenarios: UI, API, DB

- End to End Testing -> Involving Functionality  
End to End Testing -> Involving Functionality Plus Each Layer Of Application
- 1) Go to UI -> Add An Employee
  - 1) Go to DB and verify if employee is added and all data is matching
  - 2) API -> GET request and verify if employee is added successfully and all data is matching
  - > makes changes in front end and verify in database and REST API.
- 2) Go to UI -> add an employee:  
check in UI search page.
- 3) POST an employee using REST API:
  - 1) send a GET request with API and verify
  - 2) Go to DB and verify if employee is added successfully and all data is matching
  - 3) Go to front end(website) and verify that data posted is displayed
  - > makes changes using REST API then verify in DB and UI
- 4) INSERT an employee into database:
  - 1) run select statement in DB and verify what you inserted is there in tables
  - 2) send API GET request and verify JSON is matching data you inserted to DB
  - 3) Go to front end(website) and verify that data inserted to DB is displayed
  - > make changes in DB using SQL and verify in REST API & front end



# Additional notes

- Big companies use database in a very simple way. Because SQL is very slow languages. It doesn't have OOP concept. So, instead of doing manipulation in database, companies would rather to do in Web Services. Because they use Java, Python, C++ in web service layer and those are much faster than SQL.
- JDBC is an API. It enables Java to interact with the Oracle DataBase.
- Applications such as Uber and Waze pay some money to google and get the permission to access google maps API to use in their own application. This communication is provided by API.
- Or Google use other newspapers to publish news on Google news. Google simply connects to these news sources (CNN, Fox, Newyork Times) by using API.



# TYPES OF BUGS WE FACE WHEN PERFORMING API TESTING

- Issues observed when performing API testing are
  - Stress, performance, and security issues
  - Duplicate or missing functionality
  - Reliability issues
  - Improper messaging
  - Incompatible error handling mechanism
  - Multi-threaded issues



# CHALLENGES IN API TESTING

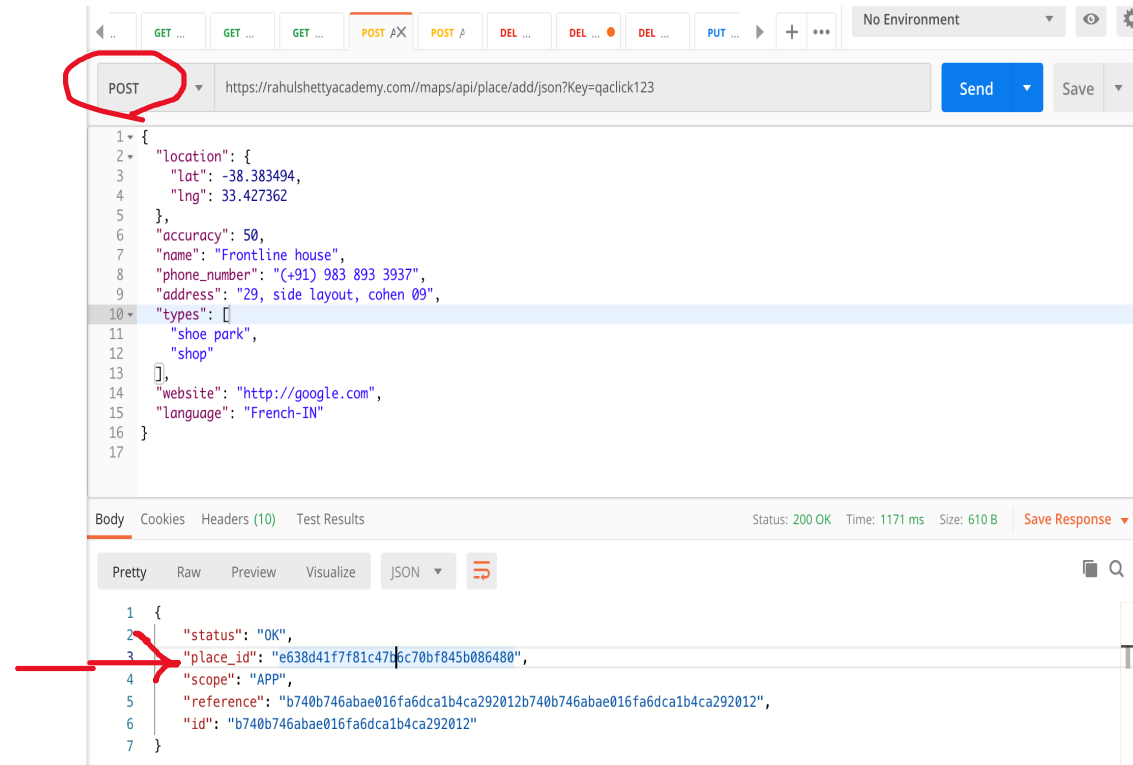
- Some of the challenges we face while doing API testing are:
  - Selecting proper parameters and its combinations
  - Categorizing the parameters properly
  - Proper call sequencing is required as this may lead to inadequate coverage in testing (Örn:DELETE'den sonra GET kullanırsan 404 gelir)
  - Verifying and validating the output
  - Due to absence of GUI it is quite difficult to provide input values



# Test flows

We need to implement the next test flow if previous flow is success:

- Single-step workflow:** Executing a single API request and checking the response accordingly. Such basic tests are the minimal building blocks we should start with, and there's no reason to continue testing if these tests fail.
- Multi-step workflow with several requests:** For example, we execute a POST request that creates a resource with id and we then use this id to check if this resource is present in the list of elements received by a GET request. Then we use a PATCH endpoint to update new data, and we again invoke a GET request to validate the new data. Finally, we DELETE that resource and use GET again to verify it no longer exists.



- Combined API and UI test:** This is mostly relevant to manual testing, where we want to ensure data integrity between the UI and API. We execute requests via the API and verify the actions through the UI or vice versa. The purpose of these integrity test flows is to ensure that although the resources are affected via different mechanisms the system still maintains expected integrity and consistent flow.

# My API testing role in my current project



- As you know I am currently working in a loan company that works with small business companies.
- Once one of our customers wants to apply for a loan from my company, he should create an account using the "get started" button and then they should fill out an online form which consists of questions such as firstName, lastName, email, phoneNumber, SSN, etc. All the information is sent to our Database. Our customer can follow and monitor the latest status of his application that is pending, accepted or denied using my loan life cycle application. In terms of the API testing, I create all the required information on behalf of the customer and using the API POST method I send them to the database. I assert that the relevant information should be also visible at the UA part. Then using API GET, PUT, DELETE method I verify that the implementation and end point are working correctly as expected and there are no bugs. For this purpose;
- 1. First, I check the API contract which is Swagger to make sure that end points are correct and to ensure that the implementation is working as specified according to API documentation (Swagger).
- 2. Then I execute my API test via the POSTMAN;
- 3. I write my Test Cases in the feature file using the Gherkin language.
- I mostly create the following *test case* groups:
  - a. Basic positive test
  - b. Extended positive testing with optional parameters and extra functionality.
  - c. Negative testing with valid input (trying to add an existing username)
  - d. Negative testing with invalid input (trying to add a username which is null)
  - e. Destructive testing (sending null, empty string, integer or other types, odd date format, deleting necessary parameters)
  - f. Security, authorization, and permission tests (sending valid or invalid access tokens to permitted or unpermitted endpoints)
- 4. For each API request I need to verify that;
  - a. I check the request and response body whether those are as written on API documentation in terms of data type and data structure.
  - b. I check HTTP status code. For example, creating a resource should return 201 CREATED and unpermitted requests should return 403 FORBIDDEN, etc.
  - c. I check Response headers. HTTP server headers have implications on both security and performance.
  - d. I check Response body. I check valid JSON body and correct field names, types, and values, including in error responses.
  - e. I check Authorization checks. I check authentication and authorization
  - f. I check Error messages. I Check the error code coverage in case API returns any error
  - g. I check Response time.

# SQL (Structured Query Language)



- [What is SQL?](#)
- [What is a Database?](#)
- [RDBMS?](#)
- [Subsets / Categories of SQL?](#)
- [SQL vs MySQL](#)
- [Database Schema](#)
- [DML and DDL Commands](#)
- [Constraints](#)
- [Primary Key and Foreign Key](#)
- [Primary Key vs UNIQUE](#)
- [Data Types in SQL](#)
- [DISTINCT](#)
- [Comparison Operators](#)
- [Where, Having, Order By, Group By](#)
- [JOINS \(All in One\)](#)
- [Inner Join](#)
- [Outer Join \(Left-Right-Full Outer Join\)](#)
- [Self Join](#)
- [Aggregate functions in SQL?](#)
- ['BETWEEN' vs 'IN' condition](#)
- [DELETE - DROP – TRUNCATE](#)
- [COMMIT – SAVEPOINT - ROLLBACK](#)

- [MetaData](#)
- [Views vs Tables](#)
- [Char vs Varchar2 data type](#)
- [Set Operators\(Union-UnionAll-Minus,Intersect\)](#)
- [Single Row functions\(Lower, Upper...\)](#)
- [Multiple Row Functions \(Count,MIN,MAX...\)](#)
- [COMMIT](#)
- [Add a COLUMN to a table](#)
- [SELECT Random Rows from a table](#)
- [Order of SQL SELECT](#)
- [Copy Data from Table1 to Table2](#)
- [Database WHITEBOX and BLACKBOX TESTING](#)
- [VIEWS](#)
- [UPDATE ROW – DELETE ROW](#)
- [CREATE TABLE](#)
- [INSERT VALUES](#)
- [UPDATE VALUE](#)
- [JDBC](#)

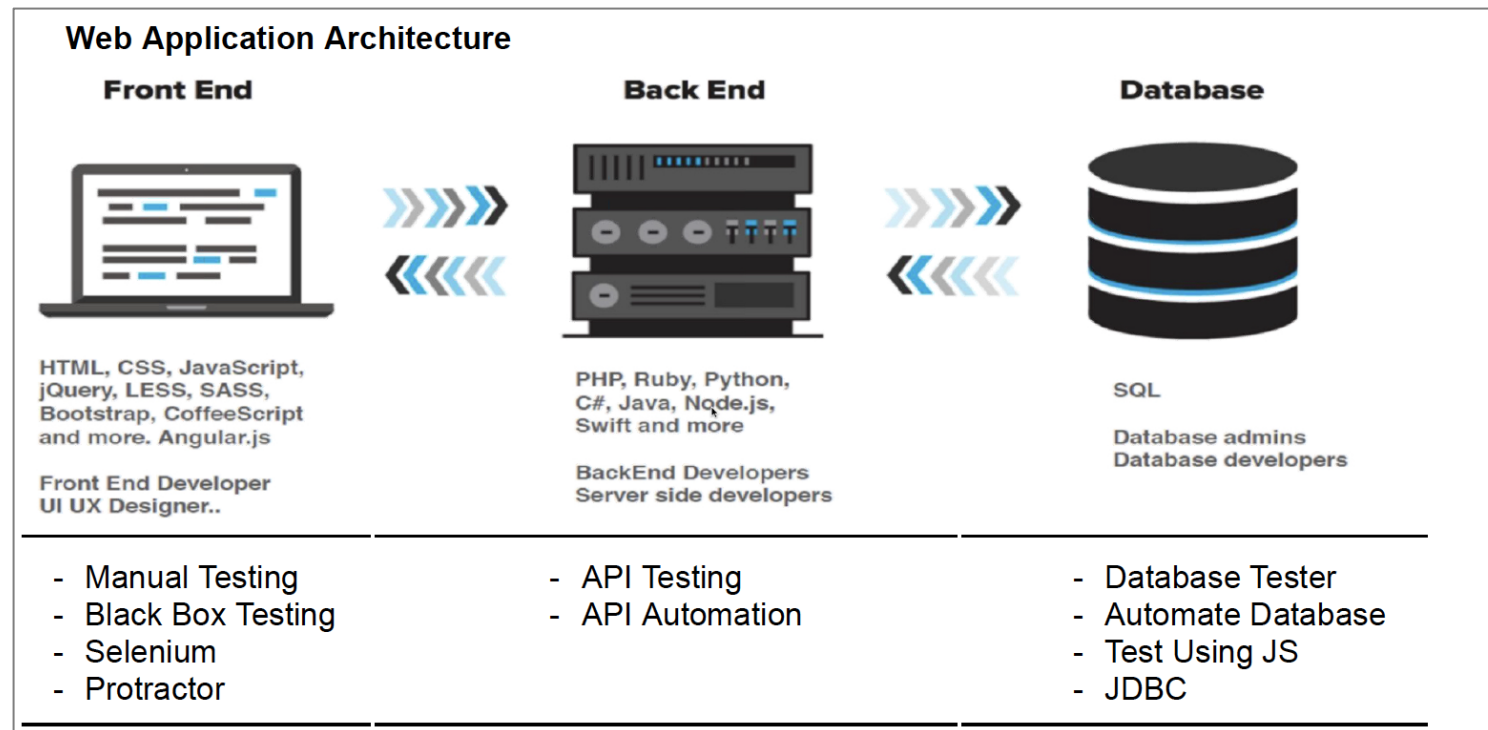
## QUERY SAMPLES

- [SELECT UNIQUE RECORDS](#)
- [1st MAX Salary](#)
- [2nd MAX Salary](#)
- [3rd MAX Salary](#)
- [MIN Salary](#)
- [Salary below the average](#)
- [Names START WITH 'A'](#)
- [MAX salary in each department](#)
- [Duplicate value](#)
- [Select Unique Records](#)
- [Current Date and Time](#)
- [SQL cheat sheet](#)

# What is SQL?



- SQL stands for Structured Query Language. It is the standard language for relational database management systems. It is especially useful in handling organized data comprised of entities (variables) and relations between different entities of the data.



# What is a Database?



- A **database** is an organized collection of data, stored and retrieved digitally from a remote or local computer system. Databases can be vast and complex, and such databases are developed using fixed design and modeling approaches.

***Database:*** is a place where data is stored in organized manner. Consists of tables. Tables are made of rows and columns. relational: Oracle 11g, MySQL, Sybase... non-relational: MongoDB

# RDBMS?



- **Relational Database Management System (RDBMS)** means that *tables in database are related using **primary/foreign key relationship**. Used to store, modify and retrieve data in the database.*
- **How are they related?**  
Primary Key (unique and not NULL)  
Foreign Key (duplicate and NULL)
- **What type of database system you have expertise with?**  
RDBMS, such as SQL and Oracle
- ***Entity Relation Model (ERM):*** *An entity can be a real-world object, that can be easily identifiable: student, class, teacher.*
- ***Attributes:*** *Things that describe the Entity. (student name, age, birthday...)*



# Subsets / Categories of SQL?



## i. DML (Data Manipulation Language)

- DML statements affect records in a table. These are basic operations we perform on data such as selecting a few records from a table, inserting new records, deleting unnecessary records, and updating/modifying existing records.

## ii. DDL (Data Definition Language)

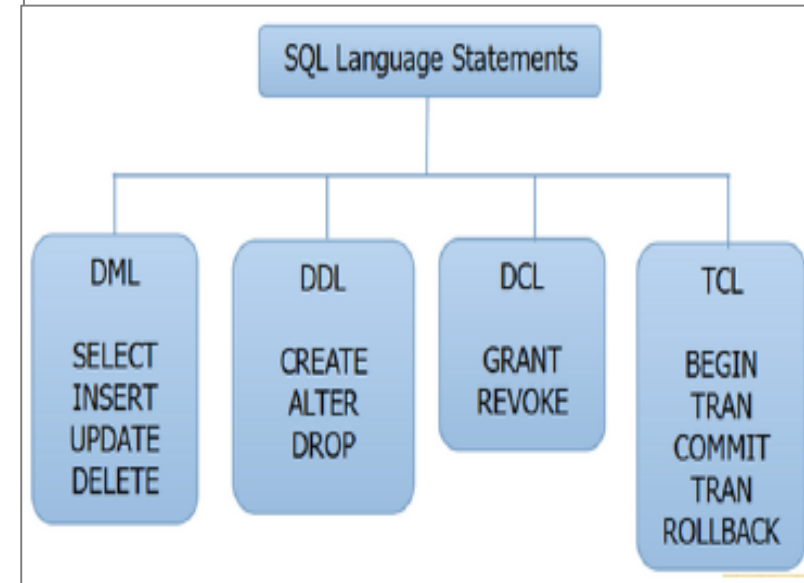
- DDL statements are used to alter/modify a database or table structure and schema. These statements handle the design and storage of database objects.

## iii. DCL (Data Control Language)

- DCL statements control the level of access that users have on database objects.

## iv. TCL (Transaction Control Language)

- TCL statements allow you to control and manage transactions to maintain the integrity of data within SQL statements.



# SQL vs MySQL

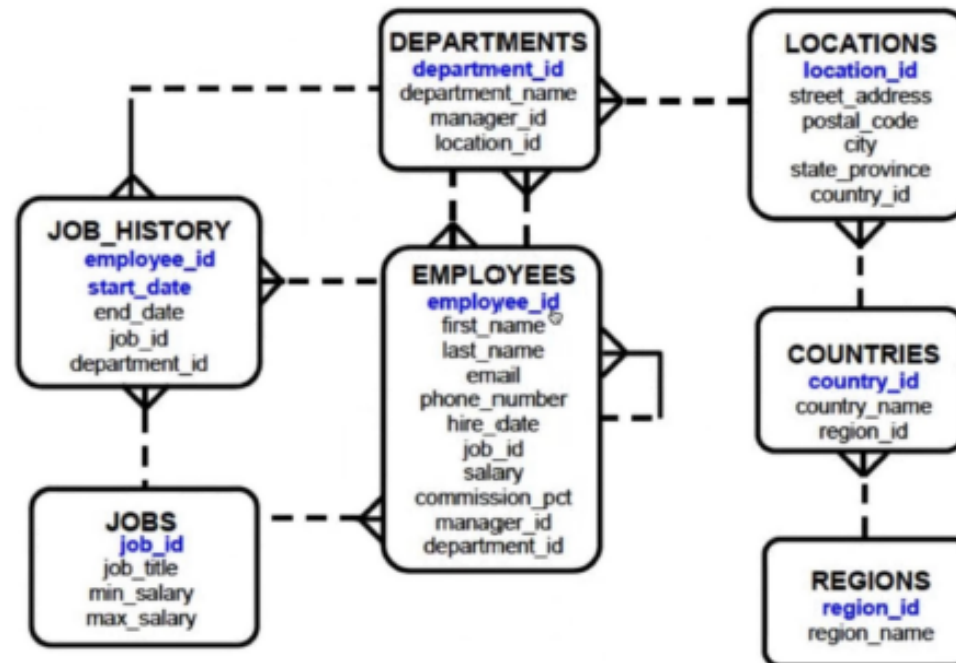


- SQL is a standard language for retrieving and manipulating structured databases. On the contrary, MySQL is a relational database management system, like SQL Server, Oracle or IBM DB2, that is used to manage SQL databases.

# DATABASE SCHEMA



- **DATABASE SCHEMA:** is a chart that shows all the tables and how they are related to each other.
- If there is no schema:
  1. Oracle ==> **SELECT** table\_name **FROM** user\_tables;
  2. MySQL ==> show tables;



# DML and DDL Commands



DML command actions can be restored.

Commands:

- **SELECT** from tablename; (read)
- **INSERT** into tablename values (...); (add)
- **UPDATE** tablename SET value WHERE location;
- **DELETE** from tablename WHERE location; (rows)
- **MERGE**

DDL command actions **cannot** be restored / undone.

Commands:

- **CREATE** table tablename (column1, column2 ...);
- **ALTER** table tablename modify value;
- **TRUNCATE** table tablename; (delete whole table data)
- **DROP TABLE**; (delete whole table with structure)
- **RENAME**
- **COMMENT**

# What are constraints?



- Properties that table column must comply with.
- Columns have constraints that defined how data can be stored.
- **Primary Key:** should be unique and NOT NULL  
**Foreign Key:** can be duplicate and NULL
- Data which is not in PK  
**Unique Key:** only unique value  
**Null:** can have null  
**Not null:** cannot have null

# Primary Key and Foreign Key



## What is Primary Key?

- It is unique column in every table in a database
- It can ONLY accept;
  - nonduplicate values
  - cannot be NULL

## What is Foreign Key?

- It is a column that comes from a different table and using Foreign key tables are related each other
- It is the primary key of another table
- It can be duplicate or null for another table

## Primary Key vs UNIQUE



- A table can have only one PRIMARY KEY whereas there can be any number of UNIQUE constrain.
- The PRIMARY KEY cannot contain null values whereas the Unique constrain can contain null values.

# Data Types in SQL



- `number(num)` - whole numbers up to `num` digits
- `number(num,num2)` - `num` whole numbers up to `num2` decimals
- `char(num)` - fixed length character/string
- `varchar2(num)` - used for varying length data
- `date` - full date
- `currency` - used for prices



# DISTINCT



- **SELECT DISTINCT** displaying non-duplicate data

```
SELECT DISTINCT job_id FROM employees;
```

# Comparison Operators



*between*

```
SELECT *
FROM employees
WHERE salary >= 4000
AND salary <= 6000;
```

```
SELECT *
FROM employees
WHERE salary BETWEEN 4000 AND 6000;
```

*is not operator <>*

```
SELECT last_name, job_id
FROM employees
WHERE job_id <> 'SA_REP';
```

```
SELECT last_name, job_id
FROM employees
WHERE NOT job_id = 'SA_REP';
```

*IN operator*

```
SELECT *
FROM employees
WHERE department_id = 90
OR department_id = 60;
```

```
SELECT *
FROM employees
WHERE department_id IN (60, 90);
```

*LIKE operator (for partial searches using wildcard '%' '\_')*

```
SELECT *
FROM employees
WHERE first_name LIKE 'N%';
```

```
SELECT *
FROM employees
WHERE last_name LIKE '_a%';
```

*top N results*

```
SELECT *
FROM employees
WHERE ROWNUM <=5;
```

*NVL (replaces NULL values with same type default value provided)*

```
SELECT NVL(commission_pct, 0)
FROM employees;
```

# WHERE, ORDER BY, GROUP BY, HAVING



- SQL clause helps to limit the result set by providing a condition to the query. A clause helps to filter the rows from the entire set of records. For example – WHERE, HAVING clause.
- **WHERE** clause in SQL is used to filter records that are necessary, based on specific conditions. WHERE is a row operation.

```
SELECT *
FROM employees
WHERE first_name LIKE 'N%';
```

- **HAVING** clause in SQL is used to filter records in combination with the GROUP BY clause. It is different from WHERE, since WHERE clause cannot filter aggregated records. HAVING is a column operation.

```
SELECT department_id, MIN (salary)
FROM employees
GROUP BY department_id
HAVING MIN (salary) < 3500;
```

- **ORDER BY** clause in SQL is used to sort the records based on some field(s) in ascending (**ASC**) or descending order (**DESC**).
- **GROUP BY** clause in SQL is used to group records with identical data and can be used in conjunction with some aggregation functions to produce summarized results from the database.

# Joins



## JOIN (INNER) JOIN

is used when retrieving data from multiple tables and will return only matching data

## LEFT (OUTER) JOIN

is used when retrieving data from multiple tables and will return left table and any matching right table records.

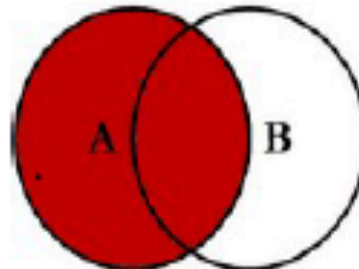
## RIGHT (OUTER) JOIN

is used when retrieving data from multiple tables and will return right table and any matching left table records.

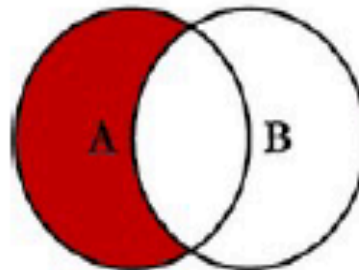
## FULL (OUTER) JOIN

is used when retrieving data from multiple tables and will return both table records, matching and non-matching

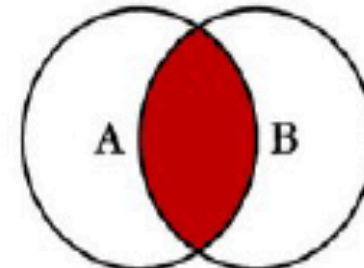
## SQL JOINS



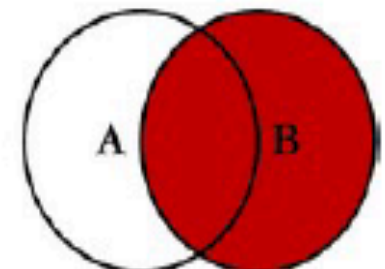
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



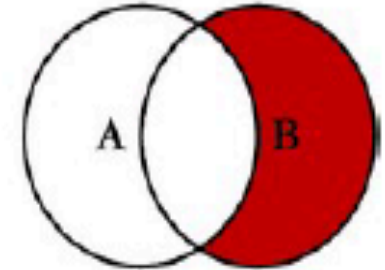
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



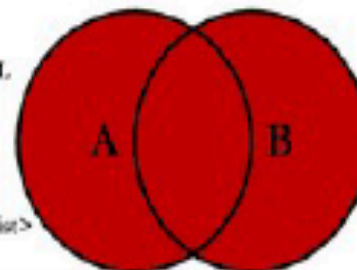
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



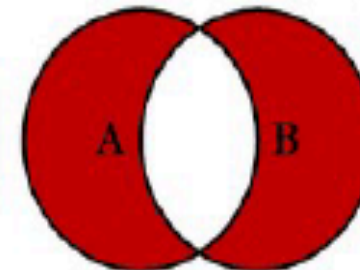
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

# INNER JOIN



**INNER JOIN** ==> used to display data from multiple tables and returns only matching records

```
SELECT employee_id, last_name, employees.department_id, department_name
FROM employees JOIN departments
ON employees.department_id = departments.department_id;
```

```
SELECT employee_id, last_name, employees.department_id, department_name
FROM employees JOIN departments
USING (department_id);
```

# OUTER JOIN (Left Outer-Right Outer-Full Outer)



2.1. **RIGHT OUTER JOIN** — used to display data from multiple tables and returns matching records and non-matching records from right hand side table

```
SELECT student_id, student_lastname, c.course_id, course_name
FROM students s RIGHT OUTER JOIN courses c -> right table
ON s.course_id = c.course_id; -> right table
```

2.2. **LEFT OUTER JOIN** — used to display data from multiple tables and returns matching records and non-matching records from left hand side table

```
SELECT student_id, student_lastname, c.course_id, course_name
FROM students s LEFT OUTER JOIN courses c -> left table
ON s.course_id = c.course_id; -> left table
```

2.3. **FULL OUTER JOIN** — display data from both tables



# SELF JOIN



**SELF JOIN** ==> | used to display data from same table

```
SELECT e1.last_name, manager_id, e2.last_name
FROM employees e1 JOIN employees e2
ON e1.manager_id = e2.employee_id;
```

# Aggregate functions in SQL?



- SQL Aggregate functions determine and calculate values from multiple columns in a table and **return a single value**.
- There are 7 aggregate functions in SQL:
  - **COUNT()** : Returns number of table rows.
  - **MAX()** : Returns the largest value among the records.
  - **MIN()** : Returns smallest value among the records.
  - **AVG()** : Returns the average value from specified columns.
  - **SUM()** : Returns the sum of specified column values.
  - **FIRST()** : Returns the first value.
  - **LAST()** : Returns last value.



# 'BETWEEN' vs 'IN' condition operators?



- BETWEEN operator is used to display rows based on a range of values in a row whereas the IN condition operator is used to check for values contained in a specific set of values.

- **Example of BETWEEN:**

```
SELECT * FROM Students
WHERE ROLL_NO BETWEEN 10 AND 50;
```

- **Example of IN:**

```
SELECT * FROM students
WHERE ROLL_NO IN (8,15,25);
```

# DELETE vs TRUNCATE vs DROP statements?

- **DELETE** is used to delete or remove one or more existing row.
- **TRUNCATE** deletes all the data from inside a table, but not the structure; cannot be retrieved back
- **DROP** deletes everything; data, table and structure from the database; cannot be retrieved back

```
//to delete values
DELETE FROM students
WHERE student_id = 104;
```

```
//clears table, keeps the table
TRUNCATE TABLE students; -> can not be rolled back
//deletes the table
DROP TABLE students; -> can not be rolled back
```

# COMMIT – SAVEPOINT – ROLLBACK



- **COMMIT** —> making all pending changes permanent;
- **SAVEPOINT** name —> marks a save point
- **ROLLBACK** —> discharges all pending changes to prev commit
- **ROLLBACK TO** name —> rolls back changes to savepoint

# Metadata



- ***Metadata: data about data. 3 types:***

- *Database Metadata*

```
DatabaseMetaData dbMetaData = connection.getMetaData();
```

- *ResultSet Metadata*

```
ResultSetMetaData rsMetaData = resultSet.getMetaData();
```

# Views vs Tables



Views	Tables
It is a virtual table that is extracted from a database.	A table is structured with a set number of columns and a boundless number of rows.
Views do not hold data themselves.	Table contains data and stores the data in databases.
A view is also utilized to query certain information contained in a few distinct tables.	A table holds fundamental client information and the cases of a characterized object.
In a view, we will get frequently queried information.	In a table, changing the information in the database changes the information that appears in the view

# CHAR and VARCHAR2 datatype in SQL



Both VARCHAR & VARCHAR2 are Oracle data types that are used to store character strings of variable length.

Their differences are:

- **VARCHAR** can store characters up to 2000 bytes while **VARCHAR2** can store up to 4000 bytes.
- **VARCHAR** will hold the space for characters defined during declaration even if all of them are not used whereas **VARCHAR2** will release the unused space.

# SET OPERATORS (UNION, UNION ALL, MINUS, INTERSECT)



1. Number of columns must be same
  2. Data type should be same
- **UNION** (returns combined rows from 2 independent queries and removes duplicates and sorts them)
  - **UNION ALL** (returns combined rows from 2 independent queries but DOES NOT remove duplicates or sort them)
  - **MINUS** (returns records from 1 query that are not present in 2 query)
  - **INTERSECT** (returns only common for both queries data)

# Single Row Functions (Lower, Upper, Substr, Length, INITCAP)



- *Single Row Functions: function that will run for each row and return value for each row*

## *LOWER*

```
SELECT LOWER (last_name)
FROM employees;
```

## *INITCAP (capital first)*

```
SELECT INITCAP (last_name)
FROM employees
```

## *INSTR (indexOf 1 based)*

```
SELECT INSTR (last_name, 'a')
FROM employees
```

## *SUBSTR*

```
SELECT SUBSTR (first_name, 0, 3)
FROM employees;
```

## *TRUNC*

```
SELECT TRUNC (commission_pct, 2)
FROM employees;
```

## *UPPER*

```
SELECT UPPER (last_name)
FROM employees;
```

## *LENGTH*

```
SELECT *
FROM employees
WHERE LENGTH (last_name) = 6;
```

## *REPLACE*

```
SELECT REPLACE(last_name, 'a', 'b')
FROM employees;
```

## *ROUND*

```
SELECT ROUND (commission_pct, 0)
FROM employees;
```

## *MOD*

```
SELECT *
FROM employees
WHERE MOD (employee_id, 2) = 0;
```



# Multiple Row Functions (Count, MIN, MAX, AVG, SUM)



*Multiple Row Functions (Group functions, Aggregate functions): will run for multiple rows and return a single value*

**COUNT** (returns number of rows)

```
SELECT COUNT (*)
FROM employees;
```

**MIN**

```
SELECT MIN (salary)
FROM employees;
```

**MAX**

```
SELECT MAX (salary)
FROM employees;
```

**AVG**

```
SELECT AVG (salary)
FROM employees;
```

**SUM**

```
SELECT SUM (salary)
FROM employees;
```

**GROUP BY** count in each department

```
SELECT department_id, COUNT (*)
FROM employees
GROUP BY department_id;
```



- **COMMIT** saves all changes made by DML statements.
- The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command.

## Add a column to a table?



- To add another column in the table, I write the command like::

```
ALTER TABLE table_name ADD column_name varchar(50);
```

# Select RANDOM ROWS from a table



- Using a **SAMPLE** clause, we can select random rows.

```
SELECT * FROM table_name SAMPLE(10);
```

# Order of SQL SELECT



Order of SQL SELECT clauses is:

- SELECT
- FROM
- WHERE
- GROUP BY
- HAVING
- ORDER BY
- Only the SELECT and FROM clauses are mandatory

## COPY Data from one table into another



- Let's assume that, we have our employee table. We have to copy this data into another table. For this purpose, we can use the **INSERT INTO SELECT** operator. Before we go ahead and do that, we would have to create another table that would have the same structure as the given table.
- First create the second table with the same table structure with copied one.
- Then use the syntax:

Let's say **employee\_duplicate** is New table

**employee** is First table that we want to copy it into new table

```
INSERT INTO employee_duplicate SELECT * FROM employee;
```



- Database **White Box testing** involves:
  - Database Consistency and ACID properties
  - Database triggers and logical views
  - Decision Coverage, Condition Coverage, and Statement Coverage
  - Database Tables, Data Model, and Database Schema
  - Referential integrity rules
- Database **Black Box testing** involves:
  - Data Mapping
  - Data stored and retrieved
  - Use of Black Box testing techniques such as Equivalence Partitioning and
  - Boundary Value Analysis (BVA)



Views are created in order to store your queries as virtual table. If there are a lot of columns and we don't want to use all columns and make the table simpler, we can create a view and reuse it repeatedly. Actually, view does not store any data however, it contains the retrieve statements to provide reusability. We can create view if we use some queries mostly.

```
create view EmployeeInfo as
select first_name || last_name as "Full Name"
from employees;
```

## Does View contain Data?

No, Views are virtual structures.



# UPDATING ROW - DELETE ROW



## UPDATING ROW

```
Update TableName set ColumnName = value where condition;
update scrumteam set firstname ='Martin' where EmployeeID='1';
update scrumteam set lastname ='Murtin' where firstname='Tom';
```

## DELETING ROW

```
delete from TableName where condition;
delete from scrumteam where firstname='Jack';
delete from scrumteam where JobTitle='SDET';
```

## CREATE TABLE



//to create a table

```
CREATE TABLE students
(
 student_id number(4) primary key,
 last_name varchar2(30) NOT NULL,
 course_id number(4) NULL);
```

//to insert values

```
INSERT INTO students VALUES (200, 'Jones', 101);
INSERT INTO students VALUES (201, 'Smith', 101);
INSERT INTO students VALUES (202, 'Lee', 102);
COMMIT; -> to save changes
```

## TO INSERT VALUES



```
//to insert values
INSERT INTO students VALUES (200, 'Jones', 101);
INSERT INTO students VALUES (201, 'Smith', 101);
INSERT INTO students VALUES (202, 'Lee', 102);
COMMIT; -> to save changes
```

## TO UPDATE VALUES



```
//to update values
UPDATE students SET course_id = 102
WHERE last_name = 'Jones'; -> if there is no condition it will update all!
```

# JDBC



- Java Database Connectivity used to connect Databases and any Java applications. Provides a call-level API for SQL-bases Database access . Connects Java Program with Database. No need to install it. Only dependency for a specific driver to use with your Database(Oracle). They are developed by DB manufacturers. Code written for one DB will work on another.
- To connect to database we need driver to work with our database; url for our db and credentials.

**Connection** —> helps connect to database

```
Connection connection = DriverManager.getConnection(url, userName, password);
```

**Statement** —> helps write and execute SQL query

```
Statement statement = connection.createStatement
(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);

// SQL Injection is a common way of hacking application databases by hackers
PreparedStatement prepStatement = connection.prepareStatement(query);
prepStatement.setInt(1, 2); -> replaces '?' In our query with number
```

**ResultSet** —> stores data that came from database

- Forward only
- Scrollable

```
ResultSet resultSet = statement.executeQuery("query");
```

**ResultSet methods:**

*next()* —> moves to next row

*previous()* —> moves to previous row

*getObject(colname/index)* —> reads data from column

*first()* —> jumps to first row

*last()* —> jumps to last row

*absolute(int row)* —> go to specific row

*beforeFirst()* —> jumps to row 0. Used to call *next()* method in while loop

*getRow()* —> returns index of current row



- SQL Query Samples

# select unique records from a table?



- You can select unique records from a table by using the **DISTINCT** keyword.

## DISTINCT

- `select distinct * from employees;` ==> retrieves any row if it has at least a single unique column.
- `select distinct first_name from employees;` ==> retrieves unique names from table. (removes duplicates)
- `select distinct count(*) from employees;` retrieve number of unique rows if any row has at least a single unique data.

# 1st MAX Salary



```
SELECT MAX(salary) FROM employees;
```

- Who is the person getting highest salary

```
SELECT first-name
FROM employees
```

```
WHERE salary = (SELECT MAX(salary) FROM employees);
```



## 2<sup>nd</sup> MAX Salary



```
SELECT first-name , salary
FROM employees
WHERE salary = (SELECT MAX(salary) FROM employees WHERE salary NOT IN
 (SELECT MAX(salary) FROM employees));
```

Or \*\*\*\*\*

```
SELECT * FROM employees
WHERE salary = (SELECT * FROM (SELECT DISTINCT salary FROM employees
 ORDER BY salary DESC) WHERE ROW NUM <=2
```

MINUS

```
SELECT * FROM (SELECT DISTINCT salary FROM employees
 ORDER BY salary DESC) WHERE ROW NUM <=1);
```

# 3rd MAX Salary



SELECT \* FROM employees

WHERE salary = (SELECT \* FROM (SELECT DISTINCT salary FROM employees  
ORDER BY salary DESC) WHERE ROW NUM <=3

MINUS

SELECT \* FROM (SELECT DISTINCT salary FROM employees  
ORDER BY salary DESC) WHERE ROW NUM <=2);

# find lowest salaries?



```
SELECT first_name, last_name, salary, job_id
FROM employees
WHERE salary = (SELECT MIN(salary) FROM employees);
```

employees whose salaries are below the average? 

```
SELECT first_name, salary FROM employees
WHERE salary <= (SELECT AVG(salary) FROM employees);
```

# find names of employee start with 'A'?



- **SELECT** \* **FROM** Employees  
**WHERE** emp\_name **LIKE** 'A%' ;

# maximum salaries in each department?



- `select deptno, Max(Sal) from emp Group By Deptno`

# duplicate names in employees?



- `SELECT first_name, COUNT (first_name) FROM employees  
GROUP BY first_name  
HAVING (COUNT(first_name) > 1);`

# select unique records from a table?



- You can select unique records from a table by using the **DISTINCT** keyword.

```
SELECT DISTINCT names FROM employees;
```



# CURRENT DATE and time in Oracle?



```
SELECT CURRENT_DATE from dual;
```

# AngularJS



- AngularJS is a JavaScript-based open-source front-end web framework.
- AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language.
- But the problem is that we spend a lot of time updating the document object model (DOM), the representation of the HTML that sits inside the browser's memory.
- That's the way for us to manipulate the HTML and code, and also to read it and see what all the HTML on the page is attributes and various values.
- So we need to do a lot of management ourselves without AngularJS, after a while, this gets really difficult to deal with and it can become just overwhelming.
- So, with AngularJS, if we could just update one side and the other side updated automatically. I mean the HTML and JS updates.
- AngularJS is attaching events and looking at variables. Every time something changes or something even could have changed, it checks all those values and then updates the page and the other variables for me.

# AngularJS VS NodeJS



- Angular JS is a front-end framework and can be used with any backend programming language like PHP, Java etc. whereas Node JS is simply a server-side language, in a web application like context it acts as a Java on the server side. ... Angular JS runs on the client browser whereas Node JS runs on the server side.

## Java Codes

<a href="#">REVERSE STRING - FOR LOOP</a>	<a href="#">FIBONACCI NUMBERS</a>
<a href="#">REVERSE STRING - STRINGBUILDER</a>	Check if a List of integers contains only odd numbers?
<a href="#">REVERSE STRING - CHAR ARRAY</a>	How to remove Whitespaces from String
<a href="#">REVERSE NUMBER</a>	How to remove leading and trailing whitespaces from a string?
<a href="#">SUM OF DIGITS</a>	Find factorial of an integer?
Swap Two Numbers Without Using a Temporary Variable	<a href="#">BINARY SEARCH</a>
With temporary variable	<a href="#">SUM OF ELEMENTS IN INTEGER ARRAY</a>
<a href="#">EVEN -ODD</a>	<a href="#">HOW TO MERGE TWO LISTS</a>
<a href="#">LARGEST NUMBER IN ARRAY</a>	Remove all occurrences of a given character from input String
<a href="#">LARGEST NUMBER IN ARRAY USING COLLECTIONS</a>	<a href="#">HOW TO CREATE ENUM</a>
<a href="#">FIND FIRST TWO NUMBER IN ARRAY</a>	Write a Program to show inheritance
<a href="#">CONVERT STRING TO ARRAY</a>	Write a Program to show try catch example
<a href="#">REMOVE EXTRA SPACES FROM STRING</a>	How do we create a Functional interface?
<a href="#">PRIME NUMBERS</a>	<a href="#">XPATH</a>
<a href="#">FIZZ BUZZ</a>	<a href="#">CSS</a>
<a href="#">STRING PALINDROME</a>	<a href="#">WAITS</a>
<a href="#">INTEGER PALINDROME</a>	Select Type Dropdown
<a href="#">FACTORIAL</a>	<a href="#">Windows/Tabs</a>
<a href="#">ONLY UNIQUE LETTERS</a>	<a href="#">Alerts/PopUps</a>
<a href="#">PYRAMID OF NUMBERS</a>	<a href="#">FRAMES / FRAMES</a>
HOW MANY ALPHA characters are present in a String?	<a href="#">ACTION CLASS</a>
<a href="#">CONVERT ARRAY TO STRING</a>	<a href="#">NESTED IFRAMES</a>
<a href="#">MAX and MIN NUMBER IN ARRAY</a>	<a href="#">Read Data From Excel File</a>

<a href="#">DUPLICATE CHARACTERS IN STRING</a>	<a href="#">SQL Server- Steps to Connect</a>
To check if a vowel is present in the string?	JavaFaker
We can upload some files from our computer to the web application using Selenium	We use browser options (Chrome or Firefox) to set default file download location and disable
To setup Log4J:	How to print all values from ArrayList

### **REVERSE STRING - FOR LOOP**

```
public static String reverseString(String str) {
 String reverse= "";
 for(int i = str.length()-1; i >=0; i--) {
 reverse += str.charAt(i); }
 return reverse; }
```

### **REVERSE STRING - STRINGBUILDER**

```
public static void main(String[] args) {
 String str = "Hello World";
 StringBuilder sb = new StringBuilder(str);
 System.out.println(sb.reverse()); }
```

### **REVERSE STRING - CHAR ARRAY**

```
public void printReverse(char[] letters, int size){
 for (int i = letters.length-1; i >= 0 ; i--){
 System.out.print(letters[i]); } }
```

### **REVERSE NUMBER**

```
public static int reverse(int number){
 int num = 1234;
 Int reversed = 0;
 int digit = 0;
 while(num != 0) {
 digit = num % 10;
 reversed = reversed * 10 + digit;
 num = num / 10; }
 System.out.println("Reversed Number: " + reversed); }
}
```

### **SUM OF DIGITS**

```
int number = 1346;
int sum = 0;
while(number > 0) {
 sum += number%10;
 number = number/10; }
System.out.println(sum); }
```

### **Swap Two Numbers Without Using A Temporary Variable**

```
a = a + b;
b = a - b;
a = a - b;
```

### **With temporary variable**

```
int temp = a;
a = b;
b = temp;
```

### **EVEN -ODD**

```
if(num % 2 == 0) {
 System.out.println(num + " is even");
} else {
 System.out.println(num + " is odd");
}
```

### **LARGEST NUMBER IN ARRAY**

```
int [] arr = {5, 6, 76, 31, 43, 1};
Arrays.sort(arr);
System.out.println(arr[arr.length-1]);
```

### **LARGEST NUMBER IN ARRAY USING COLLECTIONS**

```
public static int getLargest(int[] a, int total) {
 int temp;
 for (int i = 0; i < total; i++) {
 for (int j = i + 1; j < total; j++) {
 if (a[i] > a[j]) {
 temp = a[i];
 a[i] = a[j];
 a[j] = temp; } } }
 return a[total-1]; }
```

### **FIND FIRST TWO NUMBER IN ARRAY**

```
public void GetTwoMaxValues(int[] nums) {
 int maxOne = 0;
 int maxTwo = 0;
 Arrays.sort(nums);
 System.out.println("Max1 - " + (nums[nums.length-1]));
 System.out.println("Max2 - " + (nums[nums.length-2]));
```

### **CONVERT STRING TO ARRAY**

```
String str="YollAcademy";
char[] array=str.toCharArray();
System.out.println(array.length);
for (int i=0; i<array.length; i++) {
 System.out.println(array[i]); }
```



## **REMOVE EXTRA SPACES FROM STRING**

```
String input = "Try to remove extra spaces.";
String inputNew = input;
while (inputNew.contains(" ")) {
 inputNew = inputNew.replace(" ", "");
}
System.out.println(input);
System.out.println(inputNew);
```

## **PRIME NUMBERS**

```
public static boolean isPrime (int number) {
 for(int i=2; i < number ; i++) {
 if(number%i == 0)
 { return false; } }
 return true; }
```

## **FIZZ BUZZ**

```
public static void main(String[] args) {
 for(int i = 1 ; i<=50 ; i++) {
 if (i % (5*3) == 0) {
 System.out.println("FizzBuzz");
 } else if(i%3==0) {
 System.out.println("Fizz");
 } else if(i%5==0) {
 System.out.println("Buzz");
 } else {
 System.out.println(i); } } } }
```

## **STRING PALINDROME**

```
for(int i=word.length()-1;i>=0;i--) {
reverse+=word.charAt(i); }
if(word.equalsIgnoreCase(reverse)) {
System.out.println("The word is palindrome");
}else {
System.out.println("The word is not palindrome"); }
```

## **INTEGER PALINDROME**

```
public static boolean isPalindrome(int number) {
int palindrome = number;
int reverse=0;
while (palindrome!= 0) {
int remainder=palindrome%10;
reverse = reverse*10 + remainder;
palindrome = palindrome/10;
if (number == reverse) {
return true; }
return false;
}}}
```

## **FACTORIAL**

```
int number = 10;
int factorialSum = 1;
for(int i = 1 ; i <=number; i++) {
factorialSum = factorialSum * i; }
System.out.println("Factorial of " + number + " is " +
factorialSum); }
```

### **ONLY UNIQUE LETTERS**

```
public static String uniqueChars(String str) {
 String unique = "";
 for (int i = 0; i < str.length(); i++) {
 if (!unique.contains("" + str.charAt(i))) {
 unique += str.charAt(i); } }
 return unique; } }
```

### **PYRAMID OF NUMBERS**

```
public static void main(String[] args) {
 Scanner scan=new Scanner (System.in);
 System.out.println("How many rows you want in your
 pyramid?");
 int noOfRows=scan.nextInt();
 int rowCount=1;
 System.out.println("Here is your pyramid");
 for(int i=noOfRows;i>0;i--) {
 for(int j=1;j<=i;j++) {
 System.out.print(""); }
 for(int j=1;j<=rowCount;j++) {
 System.out.print(rowCount+ " "); }
 System.out.println();
 rowCount++; } }
```

## **How to print all values from ArrayList**

### **//Using for loop**

```
for (int i=0; i<names.size();i++) {
System.out.println(names.get(i)); }
```

### **//Using for each loop**

```
for (String value: names) {
System.out.println(value); }
```

### **//Using Iterator**

```
Iterator<String> it=names.iterator(); //create/initialize
Iterator
while(it.hasNext()) {
String name=it.next();
System.out.println(name); }
```

### **//Using while loop**

```
int count=0; while(names.size()>count) {
System.out.println(names.get(count));
count++; }
```

## **HOW MANY ALPHA characters are present in a String?**

```
String given="ghjf878997acaf9797efds&^&^*^*^";
String replaced=given.replaceAll("[a-zA-Z]", "");
int alphaChar=given.length()-replaced.length();
System.out.println(alphaChar);
```

## **CONVERT ARRAY TO STRING**

### **//1st Way**

```
System.out.println(String.valueOf(array));
```

### **//2nd Way**

```
System.out.println(Arrays.toString(array));
```

## **MAX and MIN NUMBER IN ARRAY**

### **//1st Way**

```
int[] numArray = {5, 7, 11, 15, 0, -3};
int smallest = 0;
int largest = 0;
for (int i = 0; i < numArray.length; i++) {
 if (numArray[i] > largest) {
 largest = numArray[i];
 } else {
 smallest = numArray[i]; } }
System.out.println(smallest);
System.out.println(largest);
```

### **//2nd Way**

```
Integer[] array = {7,0,25,-1,5,40};
int min = Collections.min(Arrays.asList(array));
int max = Collections.max(Arrays.asList(array));
System.out.println("Maximum value is "+max);
System.out.println("Minimum value is "+min);
```

## **DUPLICATE CHARACTERS IN STRING**

```
public static void main(String[] args) {
 String blogName = "howtodoinjava dot com";

 char[] chars = blogName.toCharArray();
 Map<Character, Integer> map = new HashMap<>();
 for(char c : chars) {
 if(map.containsKey(c)) {
 int counter = map.get(c);
 map.put(c, ++counter);
 } else {
 map.put(c, 1); } }
 System.out.println("Duplicate characters:");
 //Print duplicate characters
 for(char c : map.keySet()) {
 if(map.get(c) > 1) {
 System.out.println(c); } }
}
```

## **To check if a vowel is present in the string?**

```
public class StringContainsVowels {
 public static void main(String[] args) {

 System.out.println(stringContainsVowels("Hello")); // true

 System.out.println(stringContainsVowels("TV")); // false
 }
 public static boolean stringContainsVowels(String
input) {
 return
input.toLowerCase().matches(".*[aeiou].*"); } }
}
```

## **FIBONACCI NUMBERS**

```
public class FibonacciNumbers {
 public static int fibonacci(int n) {
 if (n <= 1)
 return n;
 return fibonacci(n - 1) + fibonacci(n - 2); }
 public static void main(String args[]) {
 int n = 10;
 System.out.println(fibonacci(n)); } }
```

## **Check if a List of integers contains only odd numbers?**

```
public static boolean onlyOddNumbers(List<Integer>
list) {
 for (int i : list) {
 if (i % 2 == 0)
 return false;
 }
 return true;
}
```

## **How to remove Whitespaces from String**

```
String removeWhiteSpaces(String input){
 StringBuilder output = new
StringBuilder();
 char[] charArray = input.toCharArray();
 for(char c : charArray) {
 if (!Character.isWhitespace(c))
 output.append(c); }
 return output.toString(); }
```

### **How to remove leading and trailing whitespaces from a string?**

```
String s = " abc def\t";
s = s.strip();
System.out.println(s);
```

### **Find factorial of an integer?**

```
public static long factorial(long n) {
 if (n == 1)
 return 1;
 else
 return (n * factorial(n - 1)); } }
```

### **BINARY SEARCH**

```
public static int binarySearch(int arr[], int low, int
high, int key) {
 int mid = (low + high) / 2;
 while (low <= high) {
 if (arr[mid] < key) {
 low = mid + 1;
 } else if (arr[mid] == key) {
 return mid;
 } else {
 high = mid - 1;
 } mid = (low + high) / 2;
 } if (low > high) {
 return -1; }
 return -1; }
```



## **SUM OF ELEMENTS IN INTEGER ARRAY**

```
int[] array = { 1, 2, 3, 4, 5 };
int sum = 0;
for (int i : array)
 sum += i;
System.out.println(sum);
```

## **HOW TO MERGE TWO LISTS**

```
List<String> list1 = new ArrayList<>();
list1.add("1");
List<String> list2 = new ArrayList<>();
list2.add("2");
List<String> mergedList = new ArrayList<>(list1);
mergedList.addAll(list2);
System.out.println(mergedList); // [1, 2]
```

## **Remove all occurrences of a given character from input String**

```
String str1 = "abcdABCDabcdABCD";

str1 = str1.replace("a", "");

System.out.println(str1); // bcdABCDBcdABCD
```

## **HOW TO CREATE ENUM**

```
public enum ThreadStates {
 START,
 RUNNING,
 WAITING,
 DEAD; }
```

### **Write a Program to show inheritance**

```
class Animal {
 String color;
}
class Cat extends Animal {
void meuw() {
System.out.println("Meuw"); }
}
```

### **Write a Program to show try catch example**

```
try {
 FileInputStream fis = new
FileInputStream("test.txt");
}catch(FileNotFoundException e) {
 e.printStackTrace(); }
```

### **How do we create a Functional interface?**

```
@FunctionalInterface interface Foo {
void test();
}
```

## **XPATH**

### **Xpath types:**

- **Absolute path** – starts with single /
- **Relative path** - starts with double // and can be anywhere on page

### **Xpath for dynamic elements**

- **Contains**
- **Using OR & AND**
- **Starts-with**
- **Text()**
- **Partial text**

### **Dynamic:**

- **`//*[contains(@name,'btn')]` → we have taken the 'name' as an attribute and 'btn' as an partial value**
- **`//label[starts-with(@id,'message')]` → This expression will identify all three elements**
- **`//*[text()='whatever']` → with text function, we find the element with exact text match**

## **CSS**

**It is faster and simpler than XPATH. But it is not as flexible as XPATH.**

- **Syntax: tag[attribute = 'value']**

- Boolean **isPresent** = `driver.findElements(By.yourLocator).size() > 0`

**This will return true if at least one element is found and false if it does not exist.**

- **isDisplayed()** method → verifies and returns a true or false based on the state of the element whether it is displayed or not. It will throw an exception if element is not present in DOM.

`driver.findElement(By. id("form_submit" )).isDisplayed() ;`

- **getText()** → method fetches text present in the html including sub-elements text. It returns value as string.

`driver.findElement(By. cssSelector(".component" )).getText() ;`

**getAttribute("")** → method gets the value of an attribute.

- in HTML code whatever is present in the left side of '=' is an attribute, value on the right side is an attribute value.

`driver.findElement (By.id("bruh")).getAttribute ("title");`

**isEnabled()** method → verifies and returns a true or false based on the state of the element whether it is enabled or not.

`.findElement (By.name("btn")).isEnabled();`

## **WAITS**

**Implicit Wait** - is set only once and lasts for the entire duration of your webdriver.

```
driver.manage().timeouts().implicitlyWait(8, TimeUnit.SECONDS);
```

**Explicit Wait** - is used to tell webdriver to wait for certain conditions or the maximum time limit before throwing an Exception.

```
Wait wait=new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.presenceOfElementLocated
(By.name("pw")));
```

**Fluent Wait** - **FluentWait** class implements the **Wait** interface in selenium.

The difference of fluent wait is that, you can define how frequently Selenium should check back if condition is met.

- **Syntax:**

- **Wait<WebDriver> fWait = new  
FluentWait<WebDriver>(driver)**
- **.withTimeout(Duration.ofSeconds(10))**
- **.pollingEvery(Duration.ofMillis(600))**
- **.ignoring(NoSuchElementException.class);**

## **Select Type Dropdown**

```
WebElement myDropdown = driver.findElement (By.id("wer"));
Select dropdown = new Select(myDropdown);
List <WebElement> allElements = dropdown. getOptions();
for (WebElement element : allElements) {
System.out. println(element.getText()); }
```

## **Windows/Tabs**

- **Since the main window we are on also has GU ID, we can get it**

- **String mainGUID = driver.getWindowHandle();**

- **We can also get window handles of all open windows tabs by selenium**

- **Set allGUIDS = driver.getWindowHandles();**

- **Once we know the GUID of a window we can switch control to it**

- **driver.switchTo().window(guid);**

- **After done you can close the new window and pass control back to main**

- **driver.close();**

- **driver.switchTo().window(mainGUID);**

## **Alerts/PopUps**

**Alerts are pop up windows that are used to get the attention of the user to perform some operation**

- **There are 3 types of alerts:**
  - **Alert**
  - **Confirmation Dialog**
  - **Prompt**
- **We cannot locate alerts using inspect tools (id, name, css, xpath etc..)**
- **Alerts are not html windows**
- **We handle alerts using:**
  - **Alert myAlert = driver.switchTo().alert();**
  - **By calling switchTo() method, we are passing control of selenium from html page to alert popup**
  - **There are 4 actions we can make with alerts**
  - **myAlert.accept(); → OK**
  - **myAlert.dismiss(); → Cancel**
  - **myAlert.getText(); → Get alert text**
  - **myAlert.sendKeys("test Text"); → Type into the prompt**

## **FRAMES / IFRAMES**

**Frame** → is an HTML tag and used to divide the same web page into various frames/windows.

**By index** → `driver.switchTo().frame(0)`

- Since the order of frames can change, it is not reliable way

- **By name or ID** → `driver.switchTo().frame("customIframe");`

- If these attributes are not present can't use it.

- **By webelement** →

- **WebElement**

`testFrame=driver.findElement(By.xpath("//*[@class='tls']"));`

- `driver.switchTo().frame(testFrame)`

## **NESTED IFRAMES**

**We need to switch to outer iframe first, and then switch to inner iframe.**

- `driver.switchTo().frame("iframe2");`

- `driver.switchTo().frame("iframe1");`

- **You cannot also go between child iframes directly. Have to go up to parent and then come down to**

**sibling. To switch control back to parent iframe:**

- `driver.switchTo().parentFrame();`

- **To switch control back to main html page**

- `driver.switchTo().defaultContent ();`



## **ACTION CLASS**

**Mouse Actions:** build(), click(), clickAndHold(), contextClick(), doubleClick(), dragAndDrop(), moveToElement(), perform(), release()

**Keyboard Actions:** sendKeys(), keyUp(), keyDown()

**In order to hover over an element, we use: moveToElement() method.**

- **Action act = new Action(driver);**

**act.moveToElement(elem).perform();**

**In order to right click an element or an webpage, we use contextClick() method**

- **Actions act = new Actions(driver);**

**act.contextClick(element).perform();**

**In order to drag and drop element we use dragAndDrop() method**

- **WebElement source = driver.findElement (By.xpath("//img"));**

- **WebElement target = driver.findElement (By.id("div2"));**

- **act.dragAndDrop(source, target). perform();**

**If we want to build series of actions methods, we use build() method**  
**.moveToElement(txtUsername)**

**.click()**

**.keyDown(txtUsername, Keys.SHIFT)**

**.sendKeys(txtUsername, "hello")**

**.keyUp(txtUsername, Keys.SHIFT)**

**.build().perform();**

**We can also use Actions class to press keys on keyboard**

- **Actions act=new Actions(driver);**
- **act.sendKeys(Keys.ENTER).build().perform();**

**There are several actions we can do with help of Javascript Executor**

- **Click an element**

- **WebElement button =driver.findElement(By.name("btnLogin"));**
- **JavascriptExecutor js = (JavascriptExecutor)driver;**
- **js.executeScript("arguments[0].click();", button);**

- **We can scroll the web page, by using JavascriptExecutor:**

- **To scroll to the a specific element**
- **js.executeScript("arguments[0].scrollIntoView(true);", element);**
- **To scroll to the end of the page:**
- **js.executeScript("window.scrollTo(0,document.body.scrollHeight)");**

## **Read Data From Excel File**

- **Create an object of FileInputStream class to read excel file**

- **FileInputStream fis = new  
FileInputStream("C:\\mydata.xlsx");**

- **Create object of XSSFWorkbook class**

- **XSSFWorkbook workbook = new XSSFWorkbook(fis);**

- **Create object of XSSFWorksheet class**

- **XSSFSheet sheet = workbook.getSheetAt(0);**

- **Access a given row:**

- **XSSFRow row = sheet .getRow(1);**

- **Access a given cell:**

- **XSSFCell cell = row.getCell(1);**

- **String value = cell.getStringCellValue();**

- **Alternatively:**

- **String value = sheet  
.getRow(1).getCell(1).getStringCellValue();**

## **SQL Server- Steps to Connect**

### **1. Load and Register the Database Driver**

**a. Depending on which database you are using: MySQL, Oracle, Postgres etc.**

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
```

### **2. Establish Connection.**

**a. Connection**

```
con=DriverManager.getConnection("jdbc:sqlserver://34.235.0.4:1433;
```

```
databaseName=MealBDb", "username", "password");
```

### **3. Create Statement Object**

**a. Statement st= con.createStatement();**

### **4. Execute the Statement**

**a. ResultSet rs= st.executeQuery("Select \* from dbo.Expenses");**

**We create an object of JavaFaker class and call methods:**

- Faker faker = new Faker();**
- String city = faker.address().city();**
- String name = faker.name().fullName();**

## **We can upload some files from our computer to the web application using Selenium**

- For file upload, we just use `sendKeys()` method and pass the directory (file path) in which our

target file is located.

- `uploadButton.sendKeys("/test-data/city.docx");`

## **We use browser options (Chrome or Firefox) to set default file download location and disable**

download popup.

- `case "chrome":`

- `WebDriverManager.chromedriver().setup();`

- `HashMap<String, Object> chromePrefs = new HashMap<String, Object>();`

- 

- `chromePrefs.put("profile.default_content_settings.popups", 0);`

- `chromePrefs.put("download.default_directory", System.getProperty("user.dir")+"/testdata/downloads/");`

- `ChromeOptions options = new ChromeOptions();`

- `options.setExperimentalOption("prefs", chromePrefs);`

- `WebDriver driver = new ChromeDriver(options);`

## **To setup Log4J:**

- 1. Add Apache Log4J dependency in the POM.XML**
- 2. Create a new log4j.properties file and save it in the Project root folder.**
- 3. Create an object of Logger class**
  - a. static final Logger oLog = LogManager.getLogger(SelectHelper.class);**
- 4. Use that object to call logger methods such as:**
  - a. oLog.info()**
  - b. oLog.warn()**
  - c. oLog.debug()**

# SQL commands

## SQL COMMANDS CHEAT SHEET

### SQL Commands

The commands in SQL are called Queries and they are of two types:

- **Data Definition Query:** The statements which defines the structure of a database, create tables, specify their keys, indexes and so on
- **Data manipulation queries:** These are the queries which can be edited.

E.g.: Select, update and insert operation

Command	Syntax	Description
ALTER table	ALTER TABLE table_name ADD column_name datatype;	It is used to add columns to a table in a database
AND	SELECT column_name(s) FROM table_name WHERE column_1 = value_1 AND column_2 = value_2;	It is an operator that is used to combine two conditions
AS	SELECT column_name AS 'Alias' FROM table_name;	It is an keyword in SQL that is used to rename a column or table using an alias name
BETWEEN	SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value_1 AND value_2;	It is an operator used to filter the result within a certain range
CASE	SELECT column_name, CASE WHEN condition THEN 'Result_1' WHEN condition THEN 'Result_2' ELSE 'Result_3' END FROM table_name;	It is a statement used to create different outputs inside a SELECT statement
COUNT	SELECT COUNT(column_name) FROM table_name;	It is a function that takes the name of a column as argument and counts the number of rows when the column is not NULL
Create TABLE	CREATE TABLE table_name ( column_1 datatype, column_2 datatype, column_3 datatype );	It is used to create a new table in a database and specify the name of the table and columns inside it

Command	Syntax	Description
GROUP BY	SELECT column_name, COUNT(*) FROM table_name GROUP BY column_name;	It is an clause in SQL used for aggregate functions in collaboration with the SELECT statement
HAVING	SELECT column_name, COUNT(*) FROM table_name GROUP BY column_name HAVING COUNT(*) > value;	It is used in SQL because the WHERE keyword cannot be used in aggregating functions
INNER JOIN	SELECT column_name(s) FROM table_1 JOIN table_2 ON table_1.column_name = table_2.column_name;	It is used to combine rows from different tables if the Join condition goes TRUE
INSERT	INSERT INTO table_name (column_1, column_2, column_3) VALUES (value_1, 'value_2', value_3);	It is used to add new rows to a table
IS NULL/ IS NOT NULL	SELECT column_name(s) FROM table_name WHERE column_name IS NULL;	It is a operator used with the WHERE clause to check for the empty values
LIKE	SELECT column_name(s) FROM table_name WHERE column_name LIKE pattern;	It is an special operator used with the WHERE clause to search for a specific pattern in a column
LIMIT	SELECT column_name(s) FROM table_name LIMIT number;	It is a clause to specify the maximum number of rows the result set must have
MAX	SELECT MAX(column_name) FROM table_name;	It is a function that takes number of columns as an argument and return the largest value among them
MIN	SELECT MIN(column_name) FROM table_name;	It is a function that takes number of columns as an argument and return the smallest value among them
OR	SELECT column_name FROM table_name WHERE column_name = value_1 OR column_name = value_2;	It is an operator that is used to filter the result set to contain only the rows where either condition is TRUE
ORDER BY	SELECT column_name FROM table_name ORDER BY column_name ASC   DESC;	It is a clause used to sort the result set by a particular column either numerically or alphabetically



Command	Syntax	Description	Commands and syntax for querying data from Single Table	Commands and syntax for querying data from Multiple Table
OUTER JOIN	SELECT column_name(s) FROM table_1 LEFT JOIN table_2 ON table_1.column_name= table_2.column_name;	It is used to combine rows from different tables even if the condition is NOT TRUE	SELECT c1 FROM t  To select the data in Column c1 from table t	SELECT c1, c2 FROM t1 INNER JOIN t2 on condition Select column c1 and c2 from table t1 and perform an inner join between t1 and t2
ROUND	SELECT ROUND(column_name, integer) FROM table_name;	It is a function that takes the column name and a integer as an argument, and rounds the values in a column to the number of decimal places specified by an integer	SELECT * FROM t  To select all rows and columns from table t	SELECT c1, c2 FROM t1 LEFT JOIN t2 on condition Select column c1 and c2 from table t1 and perform a left join between t1 and t2
SELECT	SELECT column_name FROM table_name;	It is a statement that is used to fetch data from a database	SELECT c1 FROM t WHERE c1 = 'test' To select data in column c1 from table t, where c1=test	SELECT c1, c2 FROM t1 RIGHT JOIN t2 on condition Select column c1 and c2 from table t1 and perform a right join between t1 and t2
SELECT DISTINCT	SELECT DISTINCT column_name FROM table_name;	It is used to specify that the statement is a query which returns unique values in specified columns	SELECT c1 FROM t ORDER BY c1 ASC (DESC) To select data in column c1 from table t either in ascending or descending order	SELECT c1, c2 FROM t1 FULL OUTER JOIN t2 on condition Select column c1 and c2 from table t1 and perform a full outer join between t1 and t2
SUM	SELECT SUM(column_name) FROM table_name;	It is function used to return sum of values from a particular column	SELECT c1 FROM t ORDER BY c1 LIMIT n OFFSET offset To skip the offset of rows and return the next n rows	SELECT c1, c2 FROM t1 CROSS JOIN t2 Select column c1 and c2 from table t1 and produce a Cartesian product of rows in a table
UPDATE	UPDATE table_name SET some_column = some_value WHERE some_column = some_value;	It is used to edit rows in a table	SELECT c1, aggregate(c2) FROM t GROUP BY c1 To group rows using an aggregate function	SELECT c1, c2 FROM t1, t2 Select column c1 and c2 from table t1 and produce a Cartesian product of rows in a table
WHERE	SELECT column_name(s) FROM table_name WHERE column_name operator value;	It is a clause used to filter the result set to include the rows which where the condition is TRUE	SELECT c1, aggregate(c2) FROM t GROUP BY c1 HAVING condition Group rows using an aggregate function and filter these groups using 'HAVING' clause	SELECT c1, c2 FROM t1 A INNER JOIN t2 B on condition Select column c1 and c2 from table t1 and join it to itself using INNER JOIN clause
WITH	WITH temporary_name AS ( SELECT * FROM table_name) SELECT * FROM temporary_name WHERE column_name operator value;	It is used to store the result of a particular query in a temporary table using an alias		
DELETE	DELETE FROM table_name WHERE some_column = some_value;	It is used to remove the rows from a table		
AVG	SELECT AVG(column_name) FROM table_name;	It is used to aggregate a numeric column and return its average		



**FURTHERMORE:**

**SQL Developer, SQL DBA Training Masters Program**